

THE AVIATION DEVELOPMENT ECOSYSTEM



**APPLYING DO-178C,
ARP4754A,
DO-254, & RELATED
GUIDELINES**

VANCE HILDERMAN
With 25+ Industry Experts

THE AVIATION DEVELOPMENT ECOSYSTEM

**Applying DO-178C, ARP4754A,
DO-254, & Related Guidelines**

By

Vance Hilderman

With 25+ Industry Experts

Aviation Press International

Copyright © 2021 by Vance Hilderman

All rights reserved.

No part of this book may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in reviews and certain other non-commercial uses permitted by copyright law.

ISBN: 978-1-950336-16-6 (eBook)

ISBN: 978-1-950336-17-3 (Hardcover)

This book contains information obtained from hundreds of different sources all of which were deemed by the authors to be authentic and highly regarded. Reasonable efforts have been made to publish accurate and reliable data, however the publisher and authors cannot assume responsibility for the validity of any data, materials, or the consequences of their usage. The reader is reminded that the aviation development and certification field is continually updated and no published data can be 100% accurate the following day. The author and publisher have attempted to trace copyright holders of all materials and apologize to copyright holders if permission was not obtained; if any copyright material has not been acknowledged, please write and let us know so we can correct such in any future reprint.

Aviation Press International
PO Box 811547
Los Angeles, California USA 90081

Table of Contents

[Foreword](#)

[Author's Overview & Acknowledgements](#)

[1. Introduction](#)

[2. Aviation & Avionics: Development and Certification Framework](#)

[3. Type Certificates, Technical Standard Orders, and Parts Manufacturing Approval](#)

[4. ARP4761A Aviation Safety](#)

[5. ARP4754A Aircraft & System Development](#)

[6. DO-178C Avionics Software Prelude](#)

[7. DO-178C Avionics Software](#)

[8. DO-254 Avionics Hardware](#)

[9. Aviation Requirements for Systems, Software & Hardware](#)

[10. DO-200B Aeronautical Data](#)

[11. DO-278A CNS/ATM Ground & Satellite Systems/Software](#)

[12. DO-331 Model-Based Development](#)

[13. DO-332 Object-Oriented Technology \(OOT\)](#)

[14. DO-330 Tool Qualification](#)

[15. DO-326A Aviation Cyber-Security](#)

[16. Engineering Transitions](#)

[17. Aviation Development Traceability](#)

[18. Aviation System Validation & Verification](#)

[19. DO-160 Environmental Testing](#)

[20. Quality & Process Assurance in Aviation](#)

[21. Military Aviation Compliance](#)

[22. Multi-Core Processors for Avionics Via CAST-32A](#)

[23. Aviation Development/ Certification Costs vs. Benefits](#)

[24. UAV & UAS Certification Aspects](#)

[25. Conclusion](#)

[Appendix A: Avionics Acronyms](#)

[Appendix B: Aviation System Requirements Checklist](#)

Foreword

By Woodrow Bellamy, III
(Editor In Chief, Access Intelligence: Aerospace)

*“Vance Hilderman’s ability to explain the lifecycle of planning, developing, validating, verifying, and ultimately achieving certification on safety-critical airborne software and hardware systems is evident in his latest work: **Aviation Development & Certification Ecosystem**”.*

One quote Mr. Hilderman told me in an interview a few years ago still informs my coverage today and explains how this segment of aviation will continue to evolve as an interconnected ecosystem of airplane makers, avionics suppliers, aircraft service providers and operators among others:

“Everyone wants to move up the food chain. The component producers want to make subsystems, the subsystems producers want to make full systems, the systems people want to make integrated systems. It’s incredibly dynamic.”

Vance Hilderman’s ability to explain the lifecycle of planning, developing, validating, verifying, and ultimately achieving certification on safety-critical airborne software and hardware systems is evident in his latest work. **Aviation Development & Ecosystem** is truly the capstone of his 40-year career teaching engineers who would go on to advance the concept of airborne hardware and software.

Attention new engineers reading this: There are a myriad number of new type certifications, technical standard orders and parts manufacturer approvals to develop. Tomorrow’s electric air taxis and hydrogen driven turboprops are going to require entirely new methods of measuring power and displaying it to pilots inside and remotely controlling those vehicles, as well as new human-machine graphical user interfaces. The position updates of those vehicles will need to be tracked by traditionally manned aircraft and ATM systems. And that’s just within the burgeoning electric vertical takeoff and landing (eVTOL) segment of aviation.

Artificial intelligence and machine learning are more than just buzzwords for aviation engineering circles these days, EASA specifically told me in February of last year that they expect to certify the first air transport flight control system feature AI by 2025. Here's one thing that I like to keep in mind though when I hear about or see press releases with awesome new out of this world aircraft concepts:

Today's most popular in-service aircraft flown by the most popular commercial airlines feature flight control systems must adhere to the following central principle:

A catastrophic failure event due to aircraft system failure must have a probability of occurrence lower than one in 10^{-9} — or one in a billion — for person-carrying aircraft.

That means, as it always has for aviation electronics engineers, that someone's life is attached to that every line of code, real-time operating system, data bus transmission and quality assurance process in the development of new aircraft technologies. As the industry continues to embrace never-before seen feats of engineering developments in the 2020s, use this book ***Aviation Development Ecosystem*** as a guide to the creating the future.

Over the last nine years, I have been translating technical jargon that defines advancements in aviation electronics into digestible chunks of actionable information through articles and other content for Avionics International as part of Access Intelligence LLC.

Woodrow Bellamy III

About the Author



Mr. Vance Hilderman is a 35-year avionics safety-critical engineering expert. Holding a BSEE and MBA from Gonzaga University, and a Masters in Computer Engineering from USC (Hughes Fellow), Mr. Hilderman has focused on safety-critical aviation and avionics software, safety, systems, hardware development and related technical certification solutions for his entire career. In 1990, Mr. Hilderman co-founded TekSci with Mr. Bill Hubbard which successfully became the largest independent avionics/certification software services company in the world with over 150 fulltime engineers; he served as sole CEO, President, and Chief Technical Officer for its fourteen year duration. Mr. Hilderman developed the world's first DO-178, DO-200, DO-278, and DO-254 training which engineers at 90% of the world's aviation companies have procured and has performed 170+ Gap Analysis on DO-XXX and ARP4754A/4761 to eighty-plus companies onsite in thirty countries. Mr. Hilderman has consulted for 70% of the world's largest 200 aviation development companies worldwide and his contributions are flying in most of the world's aircraft and systems relied upon daily throughout the world. He has also developed advanced technologies and solutions for ARP4761, Real-Time Operating Systems, and Multi-Core computing per FAA/EASA CAST-32A. Governments throughout the world personally request Mr. Hilderman to provide onsite aviation certification services including auditing, gap analysis, mentoring, design, and training to their personnel in the USA, Canada, Germany, Brazil, Australia, New

Zealand, Italy, UK, South Korea, Spain, Israel, and Turkey.

Mr. Hilderman was the principal founder of HighRely Inc, in 2005 and served as sole CEO, President, and Chief Technical Officer for its duration; HighRely was an aviation software/system certification company which was acquired by Atego/Artisan in 2011 with Mr. Hilderman remaining as Chief Technical Officer of Aviation/Certification Services through 2013 and retaining all copyrights to his prior works under California authorship law. Mr. Hilderman also invented and developed the first certification gap analysis for DO-178, DO-278A, DO-200, ARP4754A, and DO-254, and mentored over 75 safety-critical product companies in establishing cost-effective development for avionics projects.

Technically, Mr. Hilderman is a Systems and Software/Computer engineer, and focuses upon developing system and software requirements, design, code, and tests. He also is an expert on certification strategies for both airborne and ground-based aviation (CNS/ATM) systems. Mr. Hilderman was a scientist for Hughes Aircraft where he focused upon ground-based surveillance and radar systems with airborne interfaces. Hilderman is the principal author of the world's best selling book on avionics certification, specifically on DO-178C and DO-254 and has authored 30+ technical papers covering aviation systems and software development and certification. Professional organizations have selected him to be their sole trainer in DO-178C, DO-254, and ARP4754A including Aviation Electronics Europe, Aerospace Tech Conferences, DASC, and Society of Aerospace Engineers (SAE – the Publisher of the standards). Hilderman has personally trained over 25,000 aviation engineers worldwide, and still trains over 2,000 engineers annually in DO-XXX and ARP4754A each year while also consulting for 40+ aviation companies annually.

Mr. Hilderman believes his strengths are integrity, relationship building, communicating complex solutions, and pro-active professional leadership in motivating his company AFuzion's engineers to peak performance, all combined with continual technical innovation. Safe Skies.

Author's Overview & Acknowledgements

Writing this book seemed like such a great idea ... at the time. Like many seemingly “great ideas”, the implementation was more challenging than perceived, and there were reasons more capable persons had not followed through on a similar “great idea”: writing a single, concise book containing the most relevant information developers of tomorrow’s aviation, aircraft, and avionics need today. The real challenge was in condensing forty years of experience and thousands of pages of notes into a succinctly readable book devoid of drivel and proselytizing. Fortunately, thirty of the best senior aviation engineers I’ve had the pleasure to work with have helped to hopefully achieve that goal: this book you’re now reading.

While the “Who Flew First” debate continues (hint: Americans and our French colleagues’ opinions may differ in the efficacy of track-based launches at Kitty Hawk), everyone agrees that aviation technology is vital and its advancements amaze. Tomorrow’s eVTOL, UAM, and AI based aviation systems are a testament to today’s progress. However, aviation’s exponentially increasing systems complexity belies an equally increasing development and certification complexity. While aviation safety has been generally improving, recent multiple crash tragedies show that such increasing complexity is not easily understood or managed. Aviation projects require dozens, hundreds, and for a complex airborne or ground-based system, thousands of engineers to work simultaneously. Truly, it takes 5,000+ engineers to create a safe aircraft, but only a few to let fatal errors slip through. The only solution involves an “Aviation Development Ecosystem” which provides a playbook for each participant to understand and deploy. However, there is no official single Aviation Development Ecosystem reference guide and many agencies and groups are variously involved in defining portions thereof. Thus was born my seemingly great idea of this book: codifying an Aviation Development Ecosystem intended to be applied worldwide.

As an avionics developer, and principal founder of three of the world’s more

significant avionics and aviation development services/certification companies of the past four decades, barely a week went by without an aviation development client asking me for the location of a single repository of the information herein. When I explained that what they were looking for was as impossible to find as the fountain of youth, the common reply was, “I don’t believe it. In this day of simple access to everything digital, there simply must be a source of information, surely! I’m willing to pay – where can I buy such a book containing all this information?” Silence ...

Indeed, after several years of searching for the true fountain of youth, I’m sure the intrepid, though overly optimistic, explorer Ponce de Leon likewise would have been willing to pay for knowledge of the Fountain’s exact location. But even today we’re at the infancy of the digital age. Truly, I’ve no doubt that within a decade or two, AI and Cloud technology will have evolved to enable a quick search of the entire known digital world yielding a customized “book” clarifying all the topics contained in this book. But will that information be distilled and accurate? And in a format allowing the reader to truly understand the What, Why, and How of aviation systems development and certification?

No book can be all things to all people; humans are simply too diverse in their knowledge, needs, and capabilities. So as an author developing, gathering, soliciting, distilling, compiling, and writing the information herein, I have invoked the “80% Rule” which I cite at the beginning of my aviation development training classes provided to over 25,000 aviation students worldwide: “If 80% of this information is 80% useful to 80% of the students, then I’ve achieved 100% success as a teacher.” Best case? Hopefully. Real life? Absolutely.

This book is truly the result of thousands of hours of hard work by many people. Sometimes exciting, many times less so. But always interesting. Those contributor’s and reviewer’s knowledge spans over five hundred person-years of learning which is considerable, particularly with the realization that the commercial aviation industry itself is little over one hundred years old. A sincere “Thank You” is due each of the many persons involved in this book. And to the readers: if you can simply contribute your best efforts to making aviation just a little better, a little more predictable, and

a little safer, then that is more gratifying than a formal thank you. If you have any comments or recommendations for future issues, you can always contact me; finding my contact information will take you less than one minute but we'll make the spammers work a little harder for that.

Again, this book would not have been possible without the direct and indirect contributions of so many people. A heartfelt thank you to my family who experienced more than a few evenings and weekends sharing ideas about content and context. A large thank you to Emilie Mandic and Anna Hilderman who helped to edit early versions of these chapters. A big thank you to all the technical contributors and reviewers whose expertise is reflected within these pages. A huge thank you to 300+ clients and technical colleagues of mine who informally reviewed early chapters and provided feedback. And a very special Thank You again to the formal Chapter Reviewers, and in some cases Authors, cited below; your hundreds of comments and additions were mostly all incorporated, and this book is far better because of you.

Again, I would like to personally acknowledge and thank the many formal expert technical engineering reviewers and contributors involved in this effort: they are all pillars of modern-day aviation and this world is a better, and safer, place because of them. And a huge Thank You to my beautiful wife Colleen Hilderman, whose outer beauty is surpassed only by the inner – you are an inspiration to us all. And four of our six children (Anna, Emilie, Evan, and Dan) who helped edit or review portions of this book – you are appreciated and loved, as are the other two (Carson and Kaitlin) who somehow miraculously were otherwise engaged when early reviewing was needed.

To all the above: thank you for having spent the extra time so the readers benefit from your many inputs. It is no easy matter to succinctly clarify complex subjects and it would have been far easier to simply make yet another one thousand page technical diatribe which no actual working persons have the time or desire to read.

Safe skies,

Vance Hilderman

CEO & CTO, AFuzion Inc. www.afuzion.com

- Bachelor's Degree Electrical Engineering, *Gonzaga University*
- Master's Degree, Electrical/Computer Engineering, *University of Southern California (USC)*
- Master's Business Administration (MBA), *Gonzaga University*

Special “THANK YOU” to the formal Chapter Reviewers & Contributors cited below:

Technical Contributors, Alphabetical Order:

1. Mr. Mark Brown, Senior Architect, Lynx Software
2. Mr. Aharon David, Cyber-Security and Aviation Expert, AFuzion-Infosec
3. Mr. Jon Lynch, FAA DER, AFuzion
4. Mr. Greg Wilson, FAA DER

Technical Reviewers, Alphabetical Order:

1. Mr. Brian Allen, Staff Software Engineer, Sikorsky - a Lockheed Martin Company
2. Mr. Don Coleman, FAA DER
3. Mr. Bunyamin Coskun, Project Manager, Milsoft Inc.
4. Mr. Kevin Crozier, Senior Avionics Software Engineer, FAA Consultant DER, FAA Certified Flight Instructor and Commercial Pilot.
5. Mr. Bill De Leeuw, FAA Designee, Systems and Equipment, Software
6. Dave Deloria, Director of Engineering, Crane Aero.
7. Mr. Eric Dillaber, Development Manager, Simulink Certification & Standards
8. Dr. Bruce Douglass, Chief Evangelist, IBM Internet of Things
9. Ms. Gamze Eroglu, Former Senior Certification Specialist, STM

10. Mr. Marc GATTI, Thales AVS France SAS Scientific Director
11. Ms. Nazan Gozay Gurbuz, Taos Safety Expert, & ARP4754A/4761A Committee Member
12. Mr. Francois Guay, FAA DER
13. Mr. Alan Houp, Independent DO-160 Expert
14. Mr. Kevin Kendryna, Quality Engineering Leader, General Atomics
15. Dr. J. Lee, Avionics Manager and Colonel Korean Air Force, Retired
16. Mr. Russell McMullan (Former Technical Director – BECA)
17. Mr. George Meier, FAA DER
18. Mr. Vincenzo Mirra, Aerospace Quality & Certification Manager, Leonardo Company
19. Mr. Roger Plieske, Rohde & Schwarz
20. Mr. Bill Potter, Technical Marketing Manager, The MathWorks
21. Mr. Tom Roth, FAA DER
22. Mr. Antonio Silva (MSEE, PE), Head of Software Engineering at AgustaWestland, Compliance Verification Engineer for SW and AEH
23. Mr. Carmelo Tommasi, Software Modeling Expert, AFuzion
24. Ms. Lisa Wynn (Duncan), Software Quality Engineer, M.A., CSQE.

1

Introduction

“It was a dark and stormy night ...”

Wait. This book is about making sense of the aviation development landscape and the myriad associated “standards”. What is the pre-aviation cliché “dark and stormy night” from 1830 doing as the opening sentence here? Simple: If you have had pilot training like myself, then you know that dark and stormy nights are rarely your friend. The same is true for developers of critical aviation and avionics systems: working in a dark cockpit, and unexpected rainy-day scenarios, are never your friend. But in aviation, as in life, we can choose our good friends but never escape exposure to potentially bad influences. The difference between a good outcome and a bad outcome is based on our choices. But making good choices requires great knowledge. And experience. In aviation, there is no substitute for experience: you need it and until you have it, you are at best a “co-pilot”.

Fortunately, pilots and aviation developers alike have access to “friends” which lessen the impact of dark storms. What friends? Aircraft have increasingly advanced airborne avionics and ground-based systems, and that equipment is developed utilizing the guidance explained in this book. Combined, the darkness is illuminated and the storms tamed. Easy, right? Not at all: friends need to be close at hand to be helpful and you need to know how to work with them.

However, one challenge remains even when you have good development friends. It’s a “challenge”, not a “problem”, because challenges are problems with solutions. Quick, what’s the solution? Ahhh, we’re getting ahead of ourselves. In these days of advancing digital access and dexterity, the tendency to try for quick solutions without fully understanding the problem ecosystem permeates daily lives. First, what is the challenge? Well, that is not a simple question either because there are multiple challenges within the dark and stormy night of aviation systems development, including:

- *Every aviation system is different from every other system*

- *The Guidelines are just that: generic guidance lacking easy-to-understand objectives, clarity, or methodology*
- *There is no “one size fits all” optimal approach*
- *There is no external recipe for success, so failures of quality, cost, and schedule, are common.*

Given this non-trivial aviation landscape with multiple challenges, we need to find our friends, and quickly. We need to make sense of myriad aviation guidelines, standards, memorandums, and policies. Every aviation engineer must readily understand how the aviation development ecosystem applies to their unique situation: the aviation systems, aircraft, hardware, and software they are developing.

It's a good bet that most readers of this book have had advanced mathematics courses where similar confusion existed during early coursework. Remember those days? Students didn't want to raise their hand and ask a question, because it seemed all the other students understood; each thought they were the only one having difficulty. Now one realizes those other students were just like them: they had the same textbooks and they too were still in the dark. As an avionics and aviation professional with thirty-five years' experience consulting on 500+ aviation projects in 30 countries, I personally assure you everyone has challenges; most of them think they have problems, and very few problems have perfect solutions.

It may be the reader is smarter than their peers (after all, they acquired this book) but they still had challenges. They are also smart enough to know there is no single Rosetta Stone with which one can decipher the code to developing projects with minimum cost/schedule and still obtain maximum quality and safety. There are no free lunches, not even in government or corporate free lunch programs: someone paid. But almost everyone has competitors, and those competitors have the same challenges, perhaps even “problems”. With this book, I hope to harness the knowledge of almost four decades' experience and in the process convey a useful portion of the knowledge passed along, as I personally trained over 25,000 aviation engineers and 500+ employees how to achieve engineering and certification success. This book is about providing knowledge so you can become a better

co-pilot and ultimately a good pilot of your aviation success... possibly even Great.

What makes a technical book “good”? That is an impossibly difficult question because the answer is within the eyes of the beholder. It is a certainty that every reader of this book has different needs, different questions, and differences within the ways they learn. Indeed, it’s impossible to be all things to all people. But if 80% of the readers find that 80% of the knowledge garnered herein is 80% helpful to them, then I’d say the result was simply as good as it gets. Clearly, it would be easy to simply rehash the published guidelines, e.g. DO-178C, DO-278A, ARP-4754A, etc. Or, rehash DO-248C which is supposed to explain DO-178C. But you’ve probably already perused those documents and found them lacking. Why? They’re general-purpose guidelines whose purpose is codifying guidelines for aviation and avionics system development. What’s missing is layman explanations and integration of an overall aviation **ecosystem** relevant throughout the engineering lifecycle, provided within the context and best-practices of our real-world. That ecosystem includes the relationship and transition from aircraft and system safety to software/hardware safety. Not theoretical classroom musings, but real-world understanding because that is your world: the real world. Let’s take off on a successful learning journey by starting our engines ...

2

Aviation & Avionics: Development and Certification Framework

By Vance Hilderman

Reviewed by:

- Mr. Francois Guay, FAA DER
- Mr. George Meier, FAA DER

Aviation development is important to people for whom aviation development is important. Who are those persons? Developers, manufacturers, integrators, flight crew, airline operators and simple enthusiasts wholeheartedly agree on the importance of aviation. Who should pay attention to this book? Simple: everyone with involvement in the aviation development ecosystem. Passengers will also agree once they realize their safety is in the offing.

Hundreds of years ago, before the concept of Da Vinci's flying machine sowed the seeds of subsequent balloons, gliders, and airplanes, humans were by necessity jacks-of-all-trades. The average earthling required working knowledge of weather, farming, animal husbandry, building, and childbirth. Over time, mankind developed specialization as a pragmatic expediency: productivity improved when individuals could concentrate on their individual skills. Guilds, trade unions, governments, and regulations evolved which, for better and for worse, improved the average person's quality of life.



One outcome of the evolution toward specialization was the decoupling of usage from design knowledge. No longer did the farm implement user need to know how to design a particular farming tool; he placed some trust in the maker based upon their relationship and the price paid. Thus were sown the seeds of the modern ethos ... and then came modern airplanes.

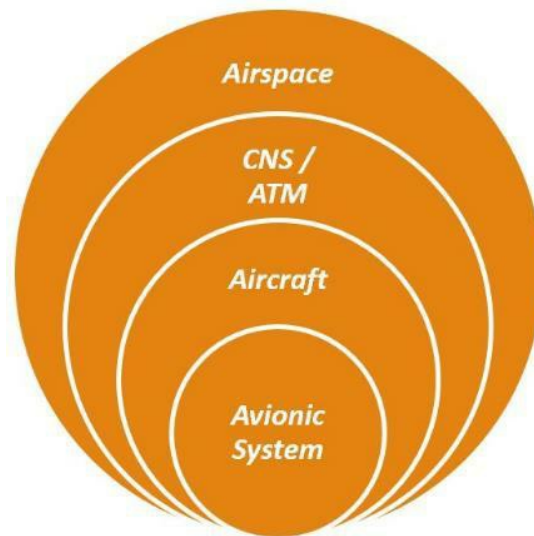
Aircraft, and flying objects in general (balloons, missiles, and unmanned aerial systems) represent hazards above (no pun intended) and beyond that of farming implements: a broken farming tool simply slows down the farming until the tool is fixed. A broken aircraft can have greater repercussions: the slowing of life can be permanent. As a result, governments and developers came together to codify safe foundations for the development and certification of aviation components and systems. The players were almost always well-intentioned but faced challenging trade-offs: over-regulate and both competitiveness and innovation would be sacrificed with the end result of decreased safety. Under-regulate and safety would be compromised with potentially disastrous results. A quandary; what to do?

Everyone wants safety if for no other reason than selfishness: everyone flies or has the potential to be struck by falling aircraft. But the aviation industry is one of most pragmatic and least selfish entities around (at least, in this author's opinion, as compared to banks, governments, lawyers, and politicians). For the aviation industry, safety is good for health AND for business. There is nothing like a bad aircraft accident to dampen flying enthusiasm and business prospects (think Concorde: one of the safest (and most beautiful) aircraft until the first ever crash in 2000 grounded the fleet permanently).

Fortunately many smart and savvy people spent decades defining a protocol for aviation safety. Whereas worldwide protocols exist for other manmade endeavors including peace-keeping (United Nations) and banking (World Bank and IMF), most would agree that aviation safety organizations are the most effective irrespective of cost. The aviation safety framework is generally well funded, science-based with a modest amount of politics, and within the western countries at least has reasonable cooperation and support of the aviation industry. This is not to say all is perfect; not at all. There are notable instances of politicking, nationalism, and some powerful concerns leveraging that power for their principal benefit. Areas of contention in aviation include CO₂ emissions, pilot work rules, noise abatement, over-flight fees, increasing congestion, airline ownership rules, and transatlantic regulation harmonization. The rise of Asia, both as an economic force and an aviation participant, is mandating additional coordination to resolve related challenges. But overall the aviation safety framework is workably intact. So

what is that framework? The following figure depicts aviation's safety layers:

- **Airspace** deals with flight planning, sector and national boundaries, and control of numerous aircraft/operations simultaneously.
- **CNS / ATM** (Communication, Navigation, Surveillance / Air Traffic Management) deals with ground and satellite-based activities relevant to aircraft and flying safety.
- **Aircraft** deals with the safety aspects pertaining to a single aircraft.
- **Avionic System** pertains to safety aspects of an avionics system where an aircraft is comprised of many such avionic systems.



As the above diagram depicts, “safety” is about the entire aviation ecosystem not merely certain aspects. Avionics developers often like to think of themselves as being on the top of the food chain. But as the above diagram clearly depicts, avionics systems are at the bottom. Why? It’s about the aviation ecosystem which is simply bigger than any one system or even any one aircraft.

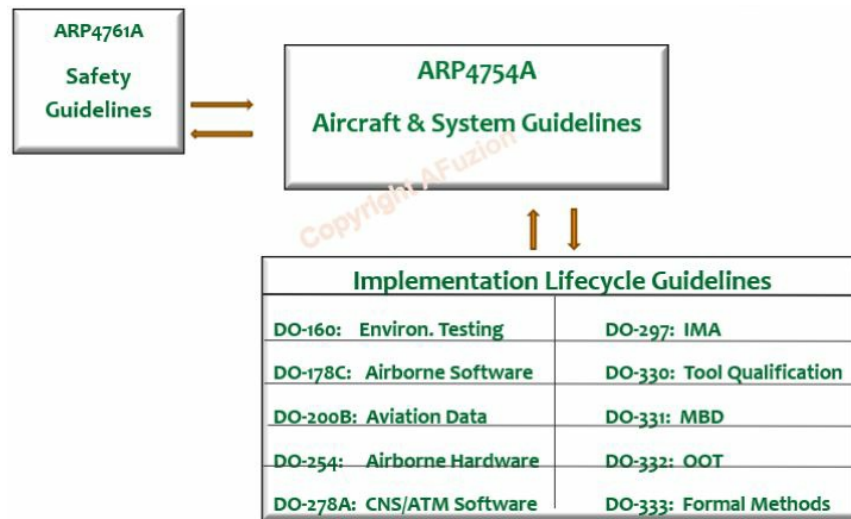
Clearly, some aspects of the aviation safety ecosystem have a greater contribution to safety than others. But if you were to ask an aviation safety expert “which aspect of aviation safety is the most important?” the answer should be “All of them”. In other words, while certain aspects can have a greater influence on safety or require more rigor, all aspects within the aviation safety ecosystem are important. But wait, that’s not the way modern

human brains operate in today's world, everything must be ranked. Think about that; have you ever caught yourself thinking about your weight, height, fitness level, hours of sleep, golf score, bank balance, calories, social media connections, or salary? Congratulations, you're normal. But aviation safety is about an ecosystem which defies simple ranking because each aspect contributes to safety and is thus individually important.

Another way to look at aviation safety is to consider a simple chain. Aviation safety resembles this chain because each link contributes to the function, and the reliability, of the chain. Want to find the chain's weakest link? Easy: simply increase tension by pulling on each end until the chain breaks. Where does it break? At its weakest link. So if a chain had one hundred links, like a modern aircraft can contain over one hundred avionics systems, the chain is only as strong as its weakest link. There is no safety benefit to having ninety-nine links of the chain extra strong with one link being weaker than necessary.



Aviation safety is about defining the “strength” or reliability of each aspect of the aviation ecosystem, with the realization that overall safety is about many different aspects and their interrelationships. Aviation safety cannot be assessed by simply stressing the system until it crashes: aviation safety is about avoiding, not causing, crashes. Therefore, aviation safety must be based upon a scientific approach which uses many techniques including analysis, engineering, process, and standards. Airspace, communications, aircraft, systems, and components are all important, but each represents a different view which must be considered within the safety evaluation process. With each of those views are numerous criteria that contribute to safety; each criterion is important though the individual contribution to safety can differ. (This leads to a discussion of Development Assurance Levels (DALs), commonly referred to as criticality levels, which will be dealt with most thoroughly in the ARP4761/A Safety section herein.) The overall aviation certification framework is depicted in the following figure:



Aviation Certification Ecosystem

As shown in the above figure, aviation certification starts in the upper left with an ARP4761/A-based safety assessment regime. The aircraft and avionics systems architectures are defined along with safety requirements which are fed into the ARP4754A-based aircraft and system development and certification process. Then, underlying avionics system aspects are dealt with via the various “DO-XXX” guidelines, where “DO” simply means North American Document. Europe has a similar ecosystem but the above documents are instead designated “ED-XXX” where “ED” means “European Document”. The remainder of this book is devoted to technical introductions and dissection of each of the above documents to explain their role within the aviation development and certification ecosystem.

Designing and certifying aviation systems has some resemblance to designing and building the links of a chain: the builder must know the purpose of the chain and how strong it is expected to be. Without that knowledge, the builder will make the link either too strong which wastes resources or too weak which causes that link to break prematurely. Either outcome is considered a failure. In avionics, the layman might think it normal to simply build it to be as close as possible to perfection with the highest possible reliability. There is no such thing as “perfection” within the realm of complex systems. Without a defined minimum level of operational reliability and integrity, there would be no way to determine if the system is “good enough”. Reliability without safeguards does not fully address product assurance and

how failures occur and what the system-level effect of such a failure is. If a chain fails, what is the impact upon the operational system? Reliability itself simply gives a failure rate: how it fails, with grace or not becomes an issue. A link in the chain may fail, but how it fails, and the impact of that failure is what causes a bad day from a safety perspective. Thus, there needs to be a dimension of integrity as we cannot simply “rely” upon reliability. Every widget designed and built is imperfect to a degree. When that combination of imperfections aligns in a “Murphy way” is what safety is supposed to identify. Therefore, avionics development becomes a study in the Avionics Development Ecosystem, e.g. this book.

In a truly free marketplace without safety-based regulations, cost and quality would continually be decreased until the link in the aviation safety chain broke, with disastrous results. Instead, avionics systems are analyzed to provide insight into the result of failures upon aircraft and human safety. Then, the relationship of that system’s correct operation to overall safety is determined and assigned a minimum threshold level for reliability: engineering techniques are then employed to ensure the necessary rigor is applied based upon that minimum reliability and integrity level.

A note on aviation development/certification cost versus safety.

Avionics development is not cheap; it’s almost always more expensive to develop products with provable quality attributes versus those without. A common worldwide misconception implies that avionics development is expensive because of the over-burdensome development process and certification considerations. There is no doubt the rigorous development and certification requirements increase initial costs; it’s impossible to be otherwise. However, that common refrain overlooks the two key questions associated with avionics development costs:

1. Given the importance of safety within aviation, are the cost increases associated with avionics development and certification excessive?
2. Are economies of scale really possible in an industry where successful projects yield hundreds of units sold versus consumer electronics where success is measured in the millions of units?

Clearly then, these two questions heavily influence the cost-versus-benefit arena if not changing the game altogether. As with politics and pizza, there will never be total agreement on these two questions. But what everyone does agree with is the following: avionics development can never be cheap, but it is possible to be cost-effective. Also, aviation is a market which perhaps more than any other places an emphasis on quality over cost: a cheap product with failures, high maintenance, or frequent replacement is neither cheap nor cost-effective. Have you ever bought a cheap tool only to be dissatisfied with its performance and replacing it with a more expensive competitor later? That situation is highly relevant in avionics, where product development costs can be in the millions of dollars and unit production costs can be in the tens of thousands.

Aircraft themselves epitomize the very definition of trade-off: cost, fuel-consumption, and payload are inextricably woven together. This explains why today there are hundreds of different new aircraft models currently for sale: each type is well suited for a particular need but incurs trade-offs versus other types. Increase payload and you increase purchase cost and operating cost. Decrease payload and you save fuel but can carry less payload with an adverse effect on revenue. And each category typically has several competitors which explains why hundreds of different aircraft models are currently for sale.

As with cost, avionics safety has a similar relationship. Additional reliability, or safety, can be added but always at additional cost. For example, developing back-up airborne satellite communication systems as if they were thrust reversers would significantly increase cost with little, if any, increase in overall safety. Again, it is a matter of trade-offs. So aviation safety blends both art and science, and the science includes a cost/benefit analysis of each proposed safety related aspect. With unlimited time and budget, safety can always be continuously improved. Consider old military aircraft with eight engines. Why eight? Because “surely” eight must be more reliable than six, which is more reliable than four, which is ... And how many engines does the world’s most modern fighter aircraft in production today have? The F-35 Joint Strike Fighter has one engine. Exactly. Meanwhile, consider the Fly-By Wire (FBW) example: conventional cable and pulley systems were replaced by FBW at what cost? Simplex (single channel) electronics and actuators

typically won't meet safety requirement, so architectures need to double, then possibly triple and sometimes quad to meet safety requirements. This geometrically increases cost, weight, complexity, and of course the cost. So within the entire avionics development ecosystem, pragmatic attention is paid to cost.

A Final Word Before Diving In ...

At the end of the day, there are no projects with unlimited time and budget. Avionics are called avionics because they must support flight and they must meet the safety framework that is explained herein. The following material provides the insights necessary to successfully navigate the cost/benefit trade-off route while still achieving sufficient safety.

3

Type Certificates, Technical Standard Orders, and Parts Manufacturing Approval

By Vance Hilderman

Reviewed by:

- Dr. J. Lee, Avionics Manager and Colonel Korean Air Force, Retired
- Mr. Antonio Silva (MSEE, PE), Head of Software Engineering at AgustaWestland, Compliance Verification Engineer for SW and AEH

So you understand that airborne electronics are part of a safe system, and guidelines exist which must be followed during the engineering lifecycle. Now you're ready to start designing the avionics system ... Not so fast! First, you need to know complete answers to the following questions (hint: the answers are embedded within an understanding of TSO's, TC's, STC's, and PMA's).

- a. Can you normally start developing an avionics system without knowing what type of aircraft it is to be installed on?
- b. Are there any specific standards which are based upon the functionality of the avionics system?
- c. If an avionics system was previously certified for one aircraft, must you always recertify it for a different type of aircraft?
- d. Can any high-quality manufacturing facility be used to produce the avionic units?

If you said “No”, “Yes”, “No”, and “No” to the above four questions, then you have at least some basic knowledge of TSO's, TC's, STC's, and PMA's. Regardless, the context for answering these fundamental questions is an important facet of understanding the avionics development ecosystem.

For commercial avionics, safety of flight is paramount. Perhaps nowhere in the avionics ecosystem is the interdependence of those safety aspects seen via an Airworthiness Pyramid with four building blocks, as depicted below:



And just like a pyramid, the airworthiness pyramid has a foundation with subsequent blocks placed upon each other. First, what is “airworthiness”?

Parts Manufacturer Approval: Combined design and production approval for modification and replacement items. It allows a manufacturer to produce and sell these articles for installation on type certificated products.

PMA

“Airworthiness” is the provable ability for an aircraft to operate safely while performing predefined functionality. When airworthiness is proven for a commercial aircraft, an airworthiness certificate may be granted. The FAA defines an airworthiness certificate to be an “FAA document which grants authorization to operate an aircraft in flight” (www.faa.gov). To receive an airworthiness certificate, FAA Order 8130 states that both of the following two conditions must be met:

- a. *The aircraft must conform to its type design. Conformity to the type design is considered attained when the aircraft configuration and the engine, propeller, and articles installed are consistent with the drawings, specifications, and other data that are part of the TC. This includes any supplemental type certificate (STC) and repairs and alterations incorporated into the aircraft.*
- b. *The aircraft must be in a condition for safe operation. This refers to the*

condition of the aircraft relative to wear and deterioration, for example, skin corrosion, window delamination/crazing, fluid leaks, and tire wear , and any outstanding airworthiness directives (AD's).

The Airworthiness Pyramid thus has the following four building blocks, summarized below with more detailed explanations to follow:

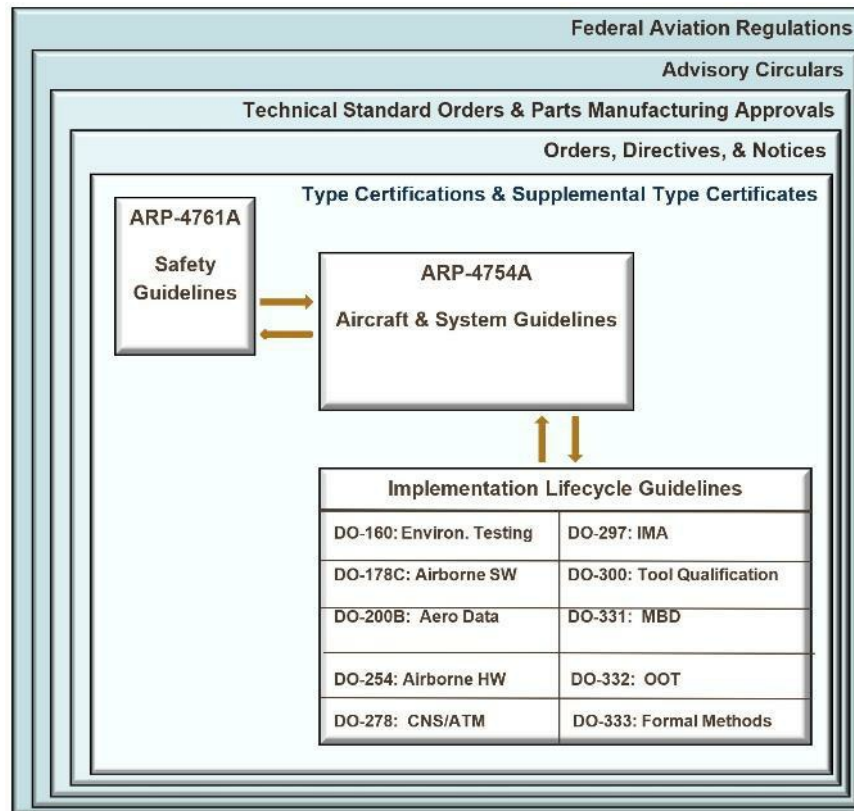
Airworthiness Pyramid So the Airworthiness Pyramid has four essential, integrated building blocks which are discussed in more detail below.



Airworthiness Pyramid

Remember: The Aviation Development & Certification Ecosystem

Remember, in avionics development it is all about the “ecosystem”. Here again is a pictorial view of the key aspects of that ecosystem. Which aspect is most important? Yes, they all are. Like a chain, the system is only as strong as the weakest link. The following figure depicts the aviation development and certification ecosystem:



The Details: TC's, STC's, TSO's, and PMA's ...

First, what is the formal regulatory basis for aircraft and avionics certification? The fundamentals are contained in Title 14 of the US Code of Federal Regulations (CFR) Aeronautics and Space, Parts:

- 21 (Certification Procedures for Products and Parts)
- 23 (Airworthiness Standards: Normal, Utility, Acrobatic, and Commuter Category Airplanes)
- 25 (Airworthiness Standards: Transport Category Airplanes)
- 27 (Airworthiness Standards: Normal Category Rotorcraft)
- 29 (Airworthiness Standards: Transport Category Rotor Craft) and the applicable amendment levels which typically define the primary basis for certification)

Type Certificate (TC).

The Type Certificate, or TC, forms the foundation of airworthiness, since it

affirms the soundness of a particular, specified manufacturing design. Harking back to the pre-digital age, a TC is just that: an actual “certificate” which is issued by a governing entity (FAA, EASA, CAA, etc.). However, a TC is more than a simple certificate as it specifies the applicable associated certification criteria which vary based upon the type of item being approved and the current regulatory requirements (including the applicable amendments to regulatory requirements). The TC includes or references sufficient information to demonstrate that the design being approved complies with the “applicable” airworthiness requirements. What are “applicable” requirements? Each type of aircraft or equipment has specific sets of requirements which apply to them. The set of requirements varies according to type, but commonly includes TSO’s, and compliance to DO-178C (software), DO-254 (airborne electronic hardware), DO-160 (environmental/EMI), etc. The TC includes a TCDS identifying the type design, operating limitations, Type Certificate Holder, Make and Model, and also specifies the applicable regulations or restrictions imposed by the certifying agency (including regulations amendment levels). For components, only engines and propellers are eligible components for TC’s. The TC is the foundation for subsequent approvals such as production and airworthiness.

There are really two key reasons Type Certificates form the foundation of airworthiness:

1. TC’s specify the applicable rules which apply and were affirmed for each item; and
2. TC’s are required prior to actually manufacturing that item.

TC Certification Plan.

As with building a house, it’s very difficult to prove compliance with building standards if the inspectors are not involved before and during the building process; that is why most buildings require a plan review prior to building, then incremental reviews as the foundation, walls, electrical, and roof are installed. Same for aircraft and avionics included as part of the TC. TC applicants must submit a detailed certification plan to the certifying agency and involve that agency or its designee throughout the project. What information is contained in a certification plan? Sufficient information to show

the applicant has a thorough understanding of the applicable regulations to enable verifiable implementation of a safe design. Specifically, the Type Certificate certification plan should detail the following information and be approved by the certification authority prior to entering the engineering implementation (actual development) phase:

- Specific information describing “who” is applying, date, and configuration identification;
- Description of design (or design change, particularly for STC (see below)) including sufficient detail in schematics or diagrams to show engineering safety;
- Operational environment, assumptions, constraints, and limitations;
- Certification basis, including applicable regulations (ARP-47XX, DO-XXX, etc.) Parts 23, 25, 27 or 29 for instance (other parts maybe applicable depending on the aircraft);
- Description of, and suitability for, showing compliance and how the configured artifacts will be used to affirm;
- Document list to be submitted which includes all required artifacts based upon the applicable regulations;
- List of test articles used to generate compliance data including any special instructions or characteristics;
- Description of how the continued operational safety requirements will be met after the TC is issued;
- Proposed project schedule showing key milestones, including all activities which require certification agency oversight or witness; and
- If FAA involved, identification of all DERs, DARs, DMIRs, and ODARs intended for use in the certification project along with areas of responsibility and capacity.

The TC applicant submits the documents to their local aviation regulatory agency and that agency provides feedback via an iterative process until the plans are approved. Then the applicant follows the plans with regular auditing of progress by the certifying agency or representatives. Just as with building a

house, inspections are incremental to allow early visibility into progress versus plan and analysis of any deviations. It should be noted that in actual practice, the submittal of plans is usually linked to the formal application for TC by the Original Equipment Manufacturer (OEM). This happens when the A/C is considered at a maturity point (verified also in experimental flights, authorized by “permit to fly” processes) giving confidence about the achievement of certification at a certain date. This is the typical process for a TC. The process for TSO/ETSO is different, since the submittal is to an ACO or EASA in front of the regulation, and it is independent from the certification of a given aircraft. A certified aircraft can be used as the “testbed” for the specific avionics being submitted.

In Europe, the submittal is done directly to the EASA proper central directorate (depending on the type of Aircraft), then EASA defines a Program Certification Management organization and the specialist Panels, one per specialty, which may include specialists from their central offices and/or applicant’s National Aviation Authority specialists. If the applicant has been granted a Design Organization Approval (DOA) and has appointed Compliance Verification Engineers (CVE) for the different specialties, EASA also defines its Level of Involvement (LOI) in the auditing process of each auditable subcomponent. The CVEs perform the auditing process in any case, with EASA specialists participation as per the LOI. An Aircraft TC is usually “validated” by national Airworthiness Authorities when sold to a Customer willing to fly in nations other than the USA or European Union. Validation between the USA and EU are facilitated by the bilateral agreements underwritten between the FAA and the EASA.

Since aviation regulations are undergoing continual revision, what happens to certification projects when the regulations change? To avoid a seemingly endless cycle of project updates to meet evolving regulations, the regulatory basis is considered “frozen” for a reasonable period of project implementation. Thus the regulatory requirements cited in the approved plans become the basis for the TC even if those regulations change before the final certification is completed (note there are exceptions to this). This usually precludes the situation where an applicant would have to change the design just because an unforeseen regulation change happened.

Design Changes.

“There are only two types of avionics designs: those that have already changed, and those that will.” (Vance Hilderman – 2014)

Commercial Aviation is barely a century old and evolution is continuous. Evolution means change. Change means some form of re-certification. What happens when a change is desired for an already-approved Type Certification (TC) design? Typically there are three possibilities:

1. Simple re-cert, where the change is considered Simple and doesn't require extensive analysis of safety implications due to the change (modification to the TC). The classification of the change is solely dependent on the potential safety effect. Example: addition of an HF radio, when keeping the standard two VHF radios active and installed. In such a case only the installation and EMC/EMI rules need to be verified: an extra HF radio is considered a “no hazard” equipment as far as its operation does not affect other equipment within the TC configuration. Note that re-cert can be done for Minor and lower changes by the TSO owner or by a TC holder with a Design Organization Approval (DOA). All other cases require re-cert with involvement of the Certification Authority.
2. Re-certification via a Supplemental Type Certificate (STC) where the change may not be simple, but it doesn't introduce new designs with associated safety considerations. Current understanding for an STC is “additive” to an existing TC, i.e., building “on top of” an approved design, typically adding functionality to avionics or “kits” to an Aircraft configuration. An STC is applicable to specific identified Aircraft serial numbers.
3. Create an entirely new design and repeat the certification process but use (leverage) many of the pre-existing artifacts (previously developed type design data). This is applicable to avionics, but not at the Aircraft level (there is only ONE TC for a given Aircraft, if you do another TC, it is by definition another aircraft type (this is why the TC is a Type Certificate).

Clearly, aviation safety would be maximized by always starting over and

creating a new design, correct? Absolutely not: starting anew simply allows new mistakes to creep into an all-new design while also becoming so expensive as to dissuade new safety-related capabilities from being incorporated. If the change is particularly simple, option #1 above is the simplest choice. If the change may introduce new safety risks not considered in the previously approved design, the certification entity may require the onerous option #3 above. But if the change can be managed within the previously approved safety context, then an STC may be used as described below.

Type Certificate Validity.

The holder of a TC must keep the TC valid by continuously following airworthiness directives (AD's), managing service bulletins, and providing technical support as required (and almost always charging for such). This is commonly referred to as "supporting the type" and often continues after the item has ceased production (and the same rules apply to Supplemental Type Certificates described below). If the TC/STC holder stops supporting the item, the certificate is relinquished back to the certifying agency and usage of that item is then disallowed for commercial aircraft operations.

Supplemental Type Certificate (STC).

An STC is a Type Certificate, but for an item which has a previously approved TC following a modification to that item. When the manufacturer shows the modification does not introduce risks not considered in the original design, that manufacturer typically requests an STC instead of an all new TC because an STC is less work than an all-new TC. The STC details the nature of the aircraft design change and how the modification affects the previously approved design including application of existing regulations. The scope of an STC can vary widely, depending upon the nature of the modification. An STC is the typical vehicle by which new avionics systems (appliances) are installed on an aircraft that has current and active TC. The aircraft installation will require the design and alteration of the aircraft so that the subject appliance can be installed on the aircraft. Like the TC process, a certification plan (Project Specific Certification Plan – PSCP) is developed and this plan defines the STC plans and certification basis. Typically STC projects follow a prescribed process: Planning (PSCP),

installation design, installation, pre-certification testing, type inspection authorization (TIA) including flight test risk assessment, installation conformity, formal (for certification credit) ground testing, and formal (for certification credit) flight testing. As part of the STC process an Experimental Airworthiness Certificate (EAC; or Permit To Fly (PTF) as it's called in Europe) may be issued prior to conducting the formal TIA, including conformity, ground and flight testing. Compliance to the regulatory requirements is substantiated by developing a series of documents to that demonstrate compliance, the following is a typical list of documents developed for an STC project:

1. STC Application
2. Project Specific Certification Plan (PSCP)
3. Conformity Plan
4. Master Data List (MDL) (DER 8110-3 approving this document)
5. Human Factors aspects of Certification Plan (HFCEP) (as required)
6. Instructions For Continued Airworthiness (ICA)
7. Airplane Flight Manual Supplement (AFMS) (DER 8110-3 recommending approval of this document)
8. Certification Compliance Matrix Check List (DER review)
9. EMI/RFI Ground Test Procedure and Plan (DER 8110-3 approving this document)
10. Ground Test Function Test Procedure and Plan (DER 8110-3 approving this document)
11. Flight Test Plan Test Procedures and Plan (DER 8110-3 recommending approval of this document)
12. Request for Type Inspection Authorization (TIA)
13. Wiring Diagram
14. Installation Instructions
15. Wire Routing
16. Wire Harness Drawing
17. Equipment Installation Diagram
18. Electrical Load Analysis
19. Special Conditions / Issue Paper (only require for unique, novel, or STC which require compliance in areas not currently controlled by FAA regulations)
20. Installation Kit List

21. Placard Drawings
22. Structural Substantiation Report
23. Weight and Balance
24. Supplemental Flight Test Report (post TIA) (DER 8110-3 approving this document)
25. Supplemental aircraft conformity inspection
26. System Safety Analysis (SSA) (aircraft level) (DER 8110-3 approving this document)
27. Mechanical / Structural Drawings (as applicable)
28. Pitot / Static Drawings (as applicable)
29. Aircraft Experimental Airworthiness Certificate aircraft owner authorization letter
30. Aircraft Experimental Airworthiness Certificate program letter
31. Application for Experimental Airworthiness Certificate (8130-6)
32. Installation Conformity and Statement of conformity (8130-9)

Typically most, if not all, of the data items listed above will need to be developed by the STC applicant in order to substantiate compliance to the STC regulatory requirements. The applicant for an STC can be the owner of one or more Aircraft, or even a separate organization, typically a major Avionics Supplier, not necessarily the same as the original TC holder. In some instances the rigor and effort required to comply with the STC certification requirements for a Part 23 aircraft (Normal, Utility, Acrobatic, and Commuter Category Airplanes) as compared to that of Part 25 aircraft (Transport Category Airplanes) is less thus requiring a reduced level of effort. Once the entire STC certification compliance package is completed and submitted to the regulatory authority the final processing of the STC can be completed and the applicant would be issued the STC.

Technical Standard Order (TSO).

A TSO (or ETSO – European TSO) comprises a unique set of performance and interface standards (MPS) for a specified item used on commercial aircraft. A “TSO Authorization” means a certification agency has reviewed and affirmed that the specified item meets the TSO MPS (design and production requirements), and the manufacturer is able to proceed with production; therefore a TSO comprises both “design” and “production”

approvals. However, depending upon the item type, a TSO is a design/production approval; a TSO is not an installation approval. The installation approval must be obtained separately; it may not necessarily imply an ability to proceed with installation on an aircraft.

An aircraft typically has dozens of different computer-based systems onboard, with large commercial aircraft containing over one hundred unique systems. Since each system performs different functionality, many of those systems must be shown to provide basic capabilities common to that type of system; these basic capabilities are thus termed “minimum performance standards”. Therefore, TSO’s embody those minimum performance and interface standards for an associated system type.

TSO’s cover many items common to civil aircraft, including avionics systems. For avionics, TSO coverage includes primary flight instruments and other systems such as: autopilots, radios, stall-warning, air data computers, AHRS/ADAHRS, windshear, flight management systems, GPS, etc. Some devices only comply with a portion of the TSO requirements; these TSO’s are referred to as partial TSOs. As the name implies, partial TSOs are for devices that provide a portion of the total TSO function. Applicants developing a partial TSO device will need to coordinate their plans with the certification authority well in advance of beginning any development work to ensure the certification authority agrees with the certification basis and partial TSO basis.

TSO’s are written by the regulatory authority and are generally supported by document orders (DO’s) developed by federal advisory committees such as RTCA. The development of DO’s includes input from a loosely integrated group of international organizations, users and the regulatory authorities. These DO’s (such as RTCA/DO-178C Software Considerations in Airborne Systems and Equipment Certification) form an integral part of the TSO requirements and are generally referenced in the TSO.

One important note: as TSO’s are important and official FAA recognition of approval, TSO’s are only extended to foreign manufacturers meeting all TSO requirements AND manufacturing in a country with a recognized bilateral agreement applicable to the avionics item in question. For example, as of this writing, the USA does not have common bilateral agreements with Turkey

whereas EASA does; therefore Turkish manufacturers can often apply for and receive ETSO's whereas obtaining a TSO (FAA) is more problematic.

Importance of TSO's.

Aside from comprising a mandatory standard for common avionic items, TSO's are important for a number of additional reasons:



Important Aspects of TSO's and Special European Considerations

TSO's then are an important layer within the Airworthiness Pyramid for avionics; they provide a common repository for additional operational characteristics for common avionics systems. For more complex, or less mature, avionics systems it is common for TSO's to be more frequently updated to thus "evolve" the minimum operational requirements for that system type. A TSO update may or may not obviate prior versions of that TSO, but updated TSO will clearly state whether or not such is the case. It should be noted that EASA is now regarding FAA TSO approvals as of limited value, not only if the TSO is outdated, but in general, since the ACO TSO process, especially for changes, is regarded sometimes as a "quick fix" way to obtain an approval; more so when the equipment provides 10% of functionality as per TSO MOPS, and 90% as extra functionality. EASA is requiring for every TSO'd equipment (and even ETSO'd) to fill in questionnaires regarding the regulations followed (AC's, DO's, etc.) explicitly, and in particular how the design assurance for functions NOT specified by the TSO has been achieved. Proceed with knowledge, and caution.

Example TSO.

To help understand TSO fundamentals, consider TSO-C63d, titled, “Airborne Weather Radar Equipment.” The title contains the acronym “TSO” so it is clearly a Technical Standard Order. The classifier “C63” is a unique designation allocated to airborne weather radar equipment. The revision letter “d” means it is the fifth revision of TSO-C63, since the second revision would have been the “a” in “C63a”. This TSO–C63d is 15 pages in length with a specified “Effective Date” of 2/28/12 meaning new weather radar certifications after February 28th, 2012 need to meet the standards specified in TSO-C63d. What does TSO-63d specify? Key contents include:

- a. **“Purpose.”** Describes what functionality is covered by this TSO, which in this case is forward looking windshear capability. It specifically excludes flight guidance system functionality which is covered elsewhere.
- b. **“Applicability.”** States that this TSO is for new airborne weather systems planned for development after the effective date (February 28, 2012) and that the prior versions of this TSO are cancelled.
- c. **“Requirements”.** Detailed enumeration of requirements covering minimum airborne weather equipment functionality, failure condition classifications, functional and environmental qualification must meet RTCA DO-173 and DO-220, Software and Hardware qualification must meet RTCA DO-178 and DO-254, data and documentation requirements,
- d. **“Minimum Performance Standards.”** Actual criteria for measuring and reporting applicable weather data for windshear detection, plus details on simulating and testing this weather data functionality.

TSO-C63d is a typical TSO in terms of its size, format, and organization. But like all TSO’s, its content is specifically tailored to meet the needs of the subject system “Weather Radar”, in this case windshear functionality contained within airborne weather radar systems.

Parts Manufacturer Approval (PMA)

PMA is a combined design and production approval for modification and replacement articles and allows a manufacturer to produce and sell these

eligible items for installation on type certificated products. PMA approval is an approval of the manufacturers Fabrication and Inspection System (FIS). A TSO-ed device does not also require the applicant to obtain PMA since the TSO is a design and production approval. PMA approval can be obtained for a replacement part utilized by a TSO device but in these cases the installer must place their modifiers nameplate on the TSO device. Parts produced under a One Time STC or Field Approval are not eligible for PMA. Standard parts, such as nuts, bolts, etc. do not require PMA.

For certificated aircraft, PMA is the basis by which normally required for devices which can affect safety to be installed on certified aircraft (this is not the case for experimental aircraft). PMA is not a general approval of inspection procedures, or materials, or processes, PMA approval is only applicable to a specific device and specific processes and procedures for that device. Manufacturers who hold a PMA can make and sell replacement parts even though they were not the original manufacturer of the aircraft or the originally installed item. PMA can be used for the production of modification parts from supplemental type certificates (STC). In these cases once an STC is issued the approval of the STC then becomes the basis for approval of a new or modified part and the PMA for that new or modified part will be issued shortly after the STC is issued.

PMA's increased in importance as air travel proliferated and evolved. Aircraft owners/operators do not want to be held hostage to a single supplier for the life of their aircraft, so they want alternate sources for parts. Conversely, certification authorities recognize that aircraft safety resembles a large, complex chain with safety only as good as the weakest link within that chain; unapproved parts could readily become that weakest link thereby affecting safety.

Obtaining a PMA normally requires two distinct and sequential activities, depicted below:



Two Steps For Obtaining PMA

Three Methods for obtaining approval of the PMA part design

Method 1. PMA by Identity by Showing Evidence of a Licensing Agreement (this is not a separate approval method, but is actually a means to substantiate identity):

Often this method is used to obtain PMA via STC licensing agreement. In this case the PMA applicant establishes a licensing agreement with the TC/STC/TSO holder and the type design data from the TC/STC/TSO is used to substantiate identity.

Method 2: PMA by Identity without a Licensing Agreement:

In this case the applicant submits a statement certifying the design is identical in all respects to an approved design. Generally, if the applicant does not have access to the original design data (in the case of complex or sophisticated parts) it is nearly impossible to obtain PMA via identity without a licensing agreement.

Method 3: PMA by Test and Computation:

This method is as it sounds, a comprehensive set of Test and Computation data demonstrating compliance with the applicable airworthiness standards.

The two sequential activities for obtaining a PMA are further described here:

Step 1: Proving Safe Design. The part to be manufactured must first be shown to have a safe design (airworthy). How is a “safe design” shown? Remember, avionics certification is about “proving your innocence” because unlike typical laws, aviation safety is predicated upon the “guilty until proven innocent” paradigm. For PMA’s, the applicant must “prove” the design is safe by:

- a. Proving all applicable certification regulations and standards have been followed for that part. Depending upon the part, for example, some combination of DO-178 (software), DO-254 (hardware), DO-160 (environmental testing), etc. may apply, along with Advisory Circulars (ACs) and possibly other DO’s and requirements.
- b. Lacking a licensing agreement per “Method 1” above, additional

Analysis can be used which shows the part is acceptable. For simple designs, this analysis could be inspection of the part and then comparison with similar approved parts to show that the new part has the same safety attributes as the already approved part. For more complex designs, analysis could be tests and calculations which directly prove the part meets all FAA safety standards. Note that for parts containing hardware or software logic, this Analysis could not be used to replace the need for DO-254 or DO-178 and this option is invalid.

Step 2: Obtaining Production Approval via Conforming QA. After Step 1 above has been accepted, the next step is to obtain actual production approval for the part. Why is this step necessary if the design has already been shown to be safe and meet all standards/regulations? Simple: consider healthy lifestyles and a person who has the “recipe book” for such which includes healthy food, exercise, stress reduction, and sleep. “Owning” such a recipe book hardly guarantees implementation – to live a healthy lifestyle requires diligence, time, money, and no junk-food binges. The same is true for safe aviation parts: the manufacturer must prove they follow the safe design and are able to manufacture it consistently while having provisions for defect detection and prevention. The process of “proof” combined with defect detection and prevention is a Quality Assurance (QA) function whereby QA proves “conformance.” This ensures that parts will be shipped to end-users only when they meet all requirements of the approved design, regulations, and standards.

PMA’s cannot be transferred and they are valid until surrendered, withdrawn, or terminated (usually by the certifying agency). PMA data can be sold (usually under license as mentioned above) but the buyer does not automatically receive permission to produce. Why? Simple: the latter two characters of “PMA” stands for “Manufacturing Approval” and it’s the very manufacturing processes, facility, and quality assurance ecosystem which must first be approved. So a buyer of PMA data, even via a license, is obligated to obtain their own PMA, though such should be easier when procuring a licensed design that previously had its own PMA. Additional notes regarding PMA’s are summarized below:



Key Notes Regarding PMA's

PMA's thus form an important part of the Airworthiness Pyramid. Since 2011, greater emphasis has been placed on a manufacturer's quality system; certification agencies such as the FAA recognize such and regularly audit for conformity as part of their ongoing certification management process.

Conclusion

The Airworthiness Pyramid provides an easy means to understand the seemingly disparate aspects of avionics type certifications. The details lie across thousands of pages of applicable rules, standards, guidelines, regulations, and orders. By following the pyramid approach described herein layer by layer, certification is no more difficult than climbing a tall, curved, ladder: time, strength, courage, and perseverance are all that is required. Enjoy the climb; the view from the top is truly outstanding.

4

ARP4761A Aviation Safety

By Vance Hilderman

With Contributions by:

- Mr. Jon Lynch

Reviewed by:

- Nazan Gozay Gurbuz, Safety Expert, & ARP4754A/4761A Committee Member

4761, 4754 ... Magic Numbers or Real Answers?

Aviation safety does not rely upon magic numbers but begins by determining real answers. Likewise, aviation safety is never an accident, but true mathematically-based Safety should prevent accidents. The numbers “4754” and “4761” are not magic but are associated with aviation safety. Safety and the Numbers have evolved – the new answers for aviation development and safety are found in 4754A and 4761A; specifically the Society of Aerospace & Automotive Engineers (SAE) ARP4754A (Aircraft & Systems Development) and ARP4761A (Safety Assessments).

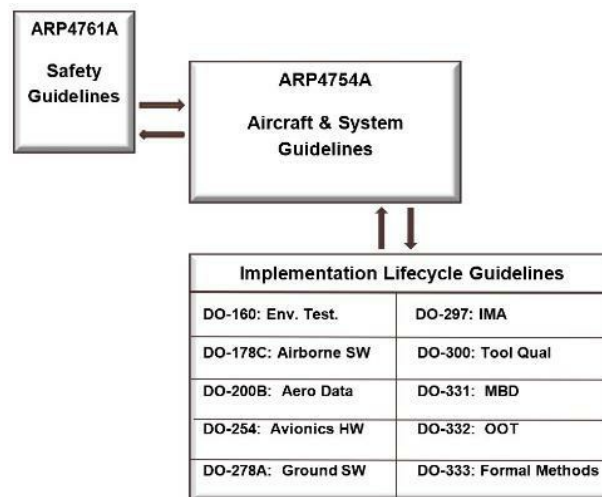
The Old Days.

In technology, the Old Days refer to any time prior to last year. In the Very Old Days (moon landings, space shuttles, new commercial jets with four engines) safety was addressed by brains and refinement: smart engineers did their best to prevent accidents then refinement was applied when those best efforts were less than perfect. In the Very Old days, computing horsepower was thinner, schedules were fatter, and acceptable safety generally ensued: space shuttles had a 98.5% chance of not exploding and commercial aircraft had fatal crashes “only” a few times annually. Then the very old days gave way to mere old days, and the very old ways didn’t work as well. Some said

the engineers were not as smart while others said those engineers were trying to make the computers too smart. But all agreed that more formalized safety was needed and the Society of Automotive Engineers (SAE) had been handling Aerospace Recommended Practices (ARP's) for decades. SAE ARP4754, *Guidelines for Development of Civil Aircraft and Systems*, was published in November 1996. Its tightly-coupled sibling ARP4761, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, was published the next month, December 1996.

ARP4761A is rather more than a guideline for aircraft safety. ARP4761A (formally issued in 2018) is officially titled “*Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*”. In fact, ARP4761 is almost a tutorial on generalized safety and how to apply various theoretical analysis to assess ongoing development activities toward aircraft safety.

Clearly, ARP4761A is tightly coupled with ARP4754A and lays the foundation for the most fundamental aspect of aircraft regulations: Safety. Clearly, viewing the aviation development and certification ecosystem, ARP4761A's prominent place in the upper left conveys its importance:



The Safety Assessment process is a vital aspect of aviation safety, and for aircraft and avionics, ARP4761 (and its newest version ARP4761A) provides the foundation.

Literally every aspect of aviation undergoes safety assessment to better

understand potential risks, quantify them, and then prevent, detect, or mitigate them. Experienced aviation persons are truthful when stating the safety assessment process is perhaps the most important element of avionics development. For avionics, the role of the safety assessment is to ensure the safety of the aircraft, its crew, and the occupants. Essentially, aircraft safety is optimized by performing careful analysis, architectural optimization, criticality level determination, component selection, architectural improvement, monitoring, and maintenance. Therefore, only by having a thorough safety assessment process can we ensure we have an architecture with additional safety-related requirements which address safety aspects. The title of ARP4761 accurately justifies its importance within this fundamental process:

“Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems & Equipment”

In an ideal world, avionics safety assessments would follow a one-size-fits-all approach consisting of uniform safety analysis steps. And in that same ideal world, humans would never be sick, gain weight, or age. Unfortunately, the ideal world cannot be assured which begets the fundamental premise of safety assessments: analyze potential problems for the real, not ideal, operating conditions. Thus, safety assessments are vital especially because aviation has no ideal world: developers of critical avionics systems spend an inordinate amount of time involved in the myriad “what if” worst-case scenarios associated with potential and realized faults.

Clearly then, if all avionics developers were perfect geniuses and could postulate and solve all possible safety scenarios in their head, they would not need a written formal safety assessment process because they performed such perfectly already. But who reviewed it? How was consistency across assessments assured? What about future systems changes? Is an undefined process really a process? And who is really an infallibly perfect genius? As the answers to these questions are all obvious, so is the need for a formal safety assessment process; ARP4761 provides the basis for that process, and at 300+ pages within that Guideline, the words here can only be a mere summary. But to understand and apply ARP4761, the practitioner needs to understand the overall Safety context and relationship to other documents.

Aviation safety is codified by a large ecosystem of information dissemination and regulations. Every individual with even an indirect role in aviation safety is governed by rules which, at a minimum, require procedures to be defined and followed. In most cases, additional regulations provide rules or guidelines covering “how” the tasks are to be performed and the minimum threshold that must be attained to achieve certification. The aviation aspects governed by safety-related regulations include:

- Aircraft pilots and crew
- Ground operators
- Maintenance personnel
- Air traffic control, communications, navigation, etc.
- Systems, hardware and software development including airborne and ground-based
- Tools including mechanical, hardware, software
- Virtually all aspects of aircraft manufacturing
- Flight testing
- Suppliers, manufacturing, updates

For avionics, the set of regulations is helpfully focused; the avionics safety assessment process primarily utilizes the following relevant industry and government documents:

- **SAE ARP4761/A** – *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*
- **SAE ARP4754A** – *Guidelines for Development of Civil Aircraft and Systems*
- **FAA Advisory Circular AC23.1309-1E** – *Equipment, Systems, and Installations in Part 23 Airplanes*
- **FAA Advisory Circular AC25.1309-1A** – *System Design and Analysis*
- **FAA Advisory Circular AC27-1B** – *Certification of Normal Category Rotorcraft*

- **FAA Advisory Circular AC29-2C** – *Certification of Transport Category Rotorcraft*
- **RTCA DO-160** – *Environmental Conditions and Test Procedures for Airborne Equipment*
- **RTCA DO-178C** – *Software Considerations in Airborne Systems and Equipment Certification*
- **RTCA DO-254** – *Design Assurance Guidance for Airborne Electronic Hardware*

If the information herein were on overall aviation safety, or even aircraft safety, the scope would expand greatly. But the purpose here is purely upon individual avionics systems. The safety assessment process utilizes experienced engineers (versed in the above documents and regulations) to proceed through a series of analyses pertaining to the system and aircraft. As the safety assessment proceeds, the following documentation typically accompanies the various analyses:

- *Aircraft Functional Hazard Assessment (FHA)*
- *Aircraft Fault Tree Analyses (FTAs)*
- *System FHAs*
- *System FTAs*
- *System Failure Modes and Effects Analyses (FMEAs)*
- *Item FTAs*
- *Item FMEAs*

A key to the above is “Traceability”: it’s imperative that relationships between aircraft operations, maintenance, systems, etc. be codified and linked: that process is termed “traceability”. In the Old Days, traceability could be handled manually or by large, institutionalized database-intensive tools. However, those manual and large-tool processes did not readily provide visibility into the relationships, safety assessments, evolution, review evidence, and derived requirements (requirements which are “derived” from a safety assessment, added to clarify a design/implementation decision, or to fill a gap).

Aviation safety involves four key safety activities as depicted in the figure

below:



The world of “safety” has its own language and lexicon. Experienced safety engineers are well versed in this language, but you are probably not reading these words if you are an experienced avionics safety engineer. To begin, please ponder the following words and test yourself by verbalizing a 1-2 sentence definition for each word (answers follow, but please spend just a few minutes pondering these words yourself before looking at the answer. Remember, you didn’t learn to drive a car by simply reading the user manual: the real learning came when you practiced driving):

- *Adverse Effect?*
- *Assessment?*
- *Average probability per flight hour?*
- *Complex System?*
- *Design Assurance Level?*
- *Extremely remote failure condition?*
- *Extremely improbable failure condition?*
- *Failure condition?*
- *Minor?*
- *Major?*
- *Hazardous?*
- *Catastrophic?*
- *Functional hazard assessment?*
- *Hazard?*
- *Primary Function?*

- *Primary System?*
- *Reliability?*
- *Secondary System?*
- *Simple System?*
- *System?*

OK, you've pondered the above words. What is their meaning when applied to avionics safety? First, there is no mandatory definition which everyone in the world adheres to or must follow. If you were to examine the dozen most common sources of aviation terminology, you would discover numerous minor differences within definitions of these terms—and some of the differences are not so minor. The various authorities and authors try to maintain consistency; in aviation this is called “harmonization”, such as when the USA's FAA and Europe's EASA coordinate to resolve differences. But your world is the real world, and that world is constantly evolving with an ever-expanding lexicon. In keeping with the coincidental but unrelated Heisenberg Principle, we should never expect perfect consensus on all definitions. So before using the following definitions, research any binding regulations which apply to your avionics activities and reach agreement with any approval agencies by citing the reference source, and definition itself, within your avionics documentation. With those WWW (wary wise words), here are what this author believes to be the most commonly accepted definitions:

Assessment: An evaluation based upon engineering judgment.

Average probability per flight hour: A representation of the number of times the subject failure condition is predicted to occur during the entire operating life of all airplanes of a type, divided by the anticipated total operating hours of all airplanes of that type.

Complex System: A system whose operation, failure modes, or failure effects are difficult to comprehend without the aid of analytical methods or structured assessment methods.

Development Assurance Level (DAL): All of those planned and systematic actions used to substantiate,

at an adequate level of confidence, that errors in requirements, design and

implementation have been identified and corrected such that the systems and items (hardware, software) satisfy the applicable certification basis. The five levels of DAL are named A, B, C, D, and E, with “A” being the highest and “E” the lowest.

Extremely remote failure condition: Those failure conditions not anticipated to occur to each airplane during its total life but which may occur a few times when considering the total operational life of all airplanes of this type.

Extremely improbable failure condition: For commuter category airplanes, those failure conditions so unlikely that they are not anticipated to occur during the entire operational life of all airplanes of one type. For other classes of airplanes, the likelihood of occurrence may be greater.

Failure condition: A condition having an effect on either the airplane or its occupants, or both, either direct or consequential, which is caused or contributed to by one or more failures or errors considering flight phase and relevant adverse operational or environmental conditions or external events. Failure conditions may be classified according to their severity as follows:

No Safety Effect: Failure conditions that would have no effect on safety (that is, failure conditions that would not affect the operational capability of the airplane or increase crew workload).

Minor: Failure conditions that would not significantly reduce airplane safety and involve crew actions that are within their capabilities. Minor failure conditions may include a slight reduction in safety margins or functional capabilities, a slight increase in crew workload (such as routine flight plan changes), or some physical discomfort to passengers or cabin crew.

Major: Failure conditions that would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions to the extent that there would be a significant reduction in safety margins or functional capabilities. In addition, the failure condition has a significant increase in crew workload or in conditions impairing crew efficiency or a discomfort to the flight crew or physical distress to passengers or cabin crew, possibly including injuries.

Hazardous: Failure conditions that would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions to the extent that there would be any of the following:

- a. A large reduction in safety margins or functional capabilities;
- b. Physical distress or higher workload such that the flight crew cannot be relied upon to perform their tasks accurately or completely; or
- c. Serious or fatal injury to an occupant other than the flight crew.

Catastrophic: Failure conditions that are expected to result in multiple fatalities of the occupants, or incapacitation or fatal injury to a flight crewmember normally with the loss of the airplane.

Functional Hazard Assessment: A systematic, comprehensive examination of functions to identify and classify Failure Conditions of those functions according to their severity

Hazard: A condition resulting from failures, external events, errors, or combinations thereof where safety is affected.

Reliability; The probability that a system or item will perform a required function under specified conditions, without failure, for a specified period of time.

Simple System: System that can be fully assured (validated and verified) by a combination of testing and analysis, relative to their requirements and identified Failure Conditions.

System: A combination of inter-related items arranged to perform a specific function(s).

A common misconception is that the safety assessment goal is to eradicate the potential for hazards; while lofty, such is impossible with complex asynchronous avionics systems. Instead, potential hazards must be effectively uncovered and quantified via the following philosophy:

- *A severe hazard may be tolerable if the probability of its occurrence is acceptably low;*
- *A probable hazard may be tolerable if the effect of the hazard on the*

aircraft, flight crew, and occupants is acceptable;

- *The probability of the occurrence of a hazard must be inversely proportional to its severity.*

The relationship between failure condition severities versus safety requirements is summarized in the following table.

Failure Condition Severities Versus Safety Requirements (Reprinted from ARP 4761)				
	MINOR	MAJOR	SEVERE MAJOR (HAZARDOUS)	CATASTROPHIC
Effect on the aircraft	<i>Slight reduction in functional capabilities or safety margins</i>	<i>Significant reduction in functional capabilities or safety margins</i>	<i>Large reduction in functional capabilities or safety margins</i>	<i>Prevents continued safe flight and landing</i>
Effect on the flight crew	<i>Slight increase in workload or use of emergency procedures</i>	<i>Physical discomfort or a significant increase in workload</i>	<i>Physical distress or excessive workload impairing ability to perform tasks</i>	<i>Fatalities or incapacitation</i>
Effect on the occupants	<i>Physical discomfort</i>	<i>Physical distress, possibly including injuries</i>	<i>Serious or fatal injury to a small number of occupants</i>	<i>Multiple fatalities</i>
Probability requirement	<i>Probable < 1.0E-3 PFH</i>	<i>Remote < 1.0E-5 PFH</i>	<i>Extremely remote < 1.0E-7 PFH</i>	<i>Extremely improbable < 1.0E-9 PFH</i>
Development Assurance Level	<i>D</i>	<i>C</i>	<i>B</i>	<i>A</i>

Although mitigation of failures is performed by setting safety qualitative and/or quantitative requirements as defined in above table, errors are mitigated by implementation of a Development Assurance Process (From DAL A to E). Systematic failure integrity is achieved by means of Development Assurance Process.

The typical safety process ‘steps’ for compliance to ARP4761A (and ARP4754A) related to these topics are:

- Identification of aircraft level and system level failure conditions
- Identification and/or selection of aircraft level or system level mitigation

strategies or operational limitations/restrictions

- Development of safety requirements to capture the failure condition mitigation strategies
- Allocation of the mitigation strategy safety requirements to the aircraft and/or systems

Each of these ‘steps’ are further described below; please note we would like to again recommend that Honda keep internal documentation evolution histories, with review records, for all of these so that downstream when FAA audits we can show actual compliance all along with primary 4754A/4761 guidance.

Identification of Aircraft Level and System Level Failure Conditions

Aircraft level FHA and system level FHA(s) should be developed. At a minimum, these FHAs should identify loss of function failure conditions as well as malfunction failure conditions. Loss of function failure conditions are sometimes referred as availability failure conditions. Malfunction failure conditions are sometimes referred to as integrity failure conditions. The following helpful notes on Availability and Integrity are from ARP4754A, which is ARP4761’s corollary document covered in the next chapter but summarized below:

ARP4754A defines availability as:

- Qualitative or quantitative attribute that a system or item is in a functioning state at a given point in time. It is sometimes expressed in terms of the probability of the system (item) not providing its output(s) (i.e. unavailability). [AFuzion Note: ‘Unavailability’ is equivalent to loss of function.]

ARP4754A defines integrity as:

- Qualitative or quantitative attribute of a system or an item indicating that it can be relied upon to work correctly. It is sometimes expressed in terms of the probability of not meeting the work correctly criteria. [AFuzion Note: ‘Not meeting the work correctly criteria’ is equivalent to

malfunction.]

Certain types of systems involve other types of failure condition classifications beyond availability and integrity. As an example, FHAs related to the required navigation performance (RNP) function usually identify failure conditions related to continuity. Continuity is defined as:

- The ability of the total system to perform its function without interruption during the intended operation. The safety concern with degraded continuity is that the system will be interrupted and not provide guidance information for the intended flight phase.

The aircraft level FHA should identify all the foreseeable loss of function (availability) and malfunction (integrity) failure conditions. System level FHAs should identify all the foreseeable availability and integrity failure conditions for the particular system under assessment. In the event that a system level FHA identifies any availability or integrity failure condition not already identified by the aircraft level FHA, such failure conditions should be flowed up to the aircraft level FHA to ensure completeness of the aircraft FHA. An SFHA is related to the AFHA through allocation of aircraft functions to system functions. AFHA failure conditions have a parent-child relationship to SFHA failure conditions. This would address the top-down traceability.

Identification and/or Selection of Aircraft Level or System Level Mitigation Strategies

Often times it is necessary to incorporate mitigation strategies into the aircraft and/or system designs so as to reduce the probability of occurrence of certain failure conditions to an acceptable level. A partial list of mitigation strategies is shown below:

- Development process rigor in accordance with the assigned FDAL or IDAL
- Flight crew annunciation of loss of function or malfunction
- Fault detection / isolation / exclusion
- Fault response
- Backup functionality
- Redundancy

- Use of high reliability (i.e., low failure rate) components
- Independence
- Partitioning/protection
- Dissimilarity/diversity between components
- Physical separation between components
- Electrical isolation between components
- Monitors (e.g., range, rate, persistence)
- Limiters
- Comparators
- Safety interlocks
- Exception handlers
- CRC/checksums
- Error checking and correcting
- Watch dog timers
- Built-in-test (power up, continuous, initiated)
- FHAs and Preliminary Safety Assessments typically identify mitigation strategies where they are necessary.

Development of Safety Requirements to Capture the Failure Condition Mitigation Strategies

Preliminary Aircraft Safety Assessments (PASA) and Preliminary System Safety Assessments (PSSA) are typically used to capture safety requirements. As part of the PASA and PSSA development, safety requirements are defined to ensure the implementation of failure condition mitigation strategies. All the safety requirements, including those related to the mitigation strategies flow into the overall suite of requirements.

Allocation of the Mitigation Strategy Safety Requirements to the Aircraft and/or Systems

The next step in the aircraft & systems development process is to allocate the requirements (including those safety requirements related to mitigation strategies) to the aircraft, systems or items. Requirements can also be allocated to various other 'elements' such as:

- Manufacturing requirements
- Maintenance requirements

- Calibration requirements
- Operational limitations or restrictions
- Flight crew training

Requirements should only be allocated to an ‘element’ where it is feasible to implement that requirement. In the event that a requirement is allocated to an ‘element’ that cannot feasibly implement the allocated requirement, a problem report (or similar mechanism) should be recorded to resolve the issue. There can be a variety of ways to resolve a requirements allocation issue such as:

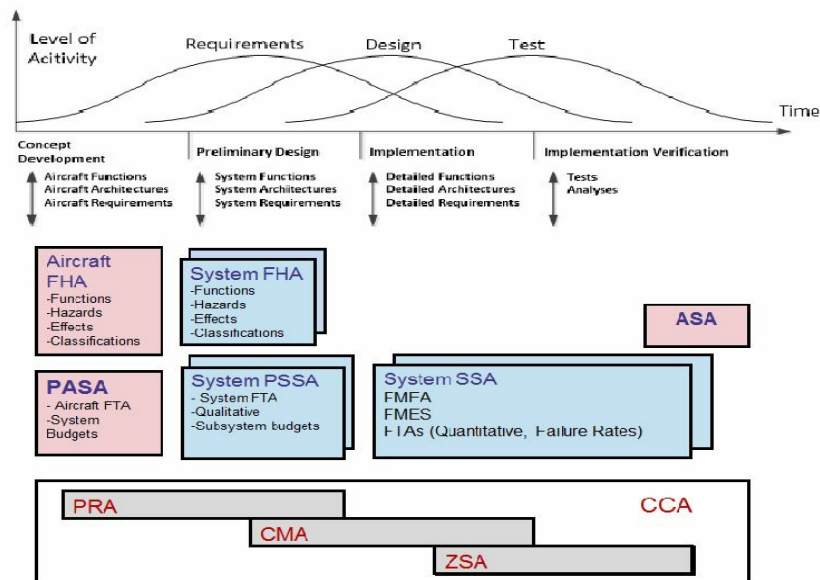
- Identification of the appropriate allocation element (e.g., reallocating a system level requirement to an aircraft level requirement)
- Modification of the allocated requirement to one that can be satisfied by the element that is assigned the requirement
- Modification of the system architecture such that the allocated requirement can be satisfied

Intrinsic Failure Conditions (Hazards)

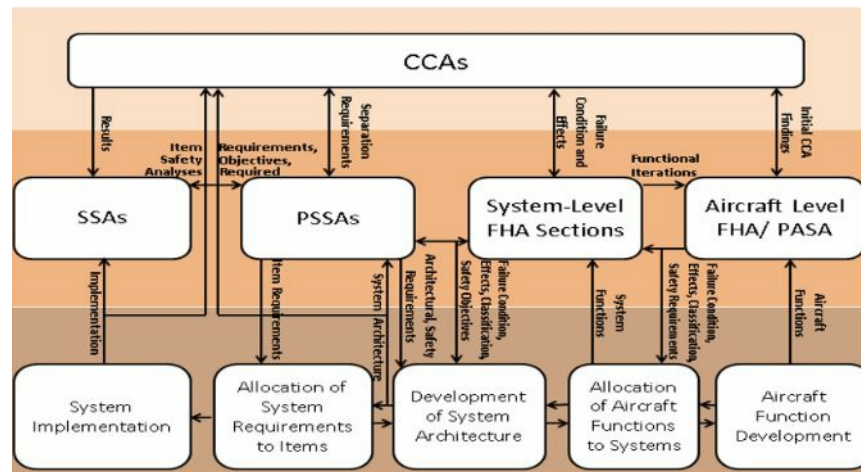
In addition to functional failure conditions, an aircraft or its equipment may contain intrinsic (physical) hazards. Aircraft intrinsic hazards are identified in a Zonal Safety Analysis (ZSA) by examining possible interference between systems in a shared aircraft zone. Equipment intrinsic hazards are identified by evaluating the bill of materials (BOM) of an assembly for properties such as hazardous materials, thermal runaways, high voltage, radiation, chemical toxicity, and explosiveness.

The following figure shows the interactions between safety assessment process activities and development process activities. The safety assessment process begins with the concept development phase and ends with the verification that the design meets the safety requirements. Because of the iterative nature of the design process, changes are made and the modified design should be reassessed in terms of safety. The “iterative approach” within ARP4754 was expanded in ARP4754A (in conjunction with ARP4761) to ensure that developers apply a formal iteration process to safety whereby safety is continually addressed with refined safety assessments

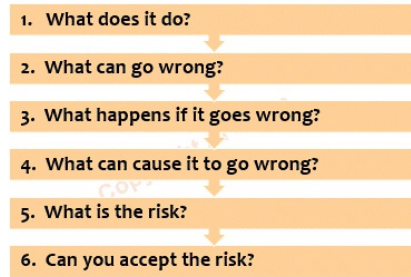
throughout the project.



The sequence of the above safety activities is depicted in the following figure; note that this figure begins in the bottom right corner and works its way upward and to the left:



The safety assessment should answer the following questions for the aircraft, then each system, as depicted below:



A key is remembering that the safety assessment process begins with an overall Functional Hazard Assessment, followed by a Preliminary Assessment; both of these are top-down. Then, during and after implementation, the final System Safety Assessment is performed inclusive of FMEA which is bottom-up. This process is depicted below:

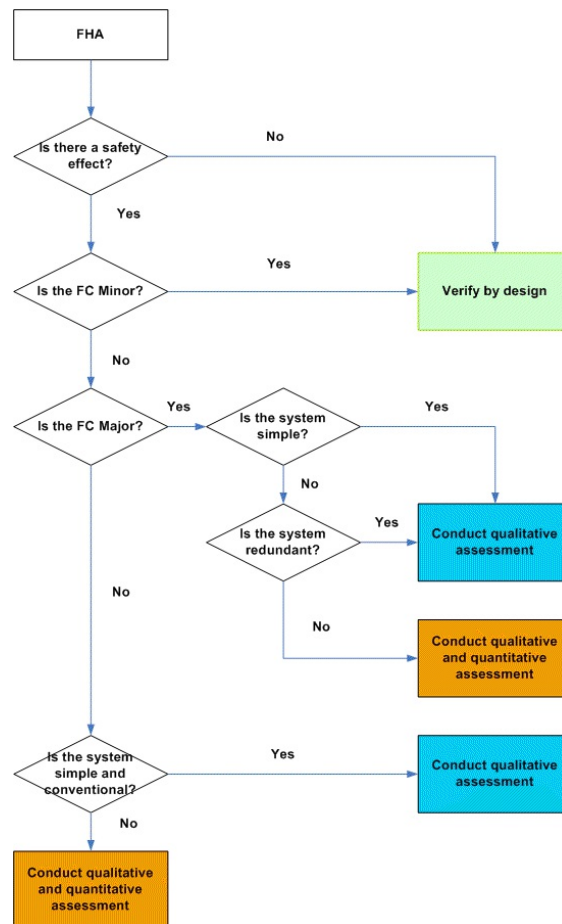


The functional hazard assessments (FHAs) are carried out at both the aircraft and system levels. The objective of the FHA is to identify failure conditions of aircraft and system functions (loss of function, malfunction, etc.), and their classification (catastrophic, hazardous, major, etc.) so that aircraft and system designs may be proposed and achieved which decrease the probability of the occurrence of the failure conditions to acceptably lesser levels. In avionics certifications, all parties recognize the importance of the FHA. The applicant is responsible for identifying each failure condition and choosing the methods for safety assessment. The applicant should then obtain early concurrence from the cognizant certifying authority on the identification of failure conditions, their classifications, and the choice of an acceptable means of compliance.

The purpose then of the FHA is summarized in the figure below:



The following figure provides an overview of the information flow to conduct safety assessments.



Functional hazard assessment (FHA):

1. Before an applicant proceeds with a detailed safety assessment, an FHA of the airplane and system functions to determine the need for and the scope of subsequent analysis should be prepared. This assessment may be conducted using service experience, engineering and operational judgment, or service experience and a top-down deductive qualitative

examination of each function. An FHA is a systematic, comprehensive examination of airplane and system functions to identify potential no safety effect, minor, major, hazardous, and catastrophic failure conditions that may arise, not only as a result of malfunctions or failure to function but also as a result of normal responses to unusual or abnormal external factors. The FHA concerns the operational vulnerabilities of systems rather than a detailed analysis of the actual implementation.

2. Each system function should be examined regarding the other functions performed by the system because the loss or malfunction of all functions performed by the system may result in a more severe failure condition than the loss of a single function. In addition, each system function should be examined regarding functions performed by other airplane systems because the loss or malfunction of different but related functions, provided by separate systems, may affect the severity of failure conditions postulated for a particular system.
3. The FHA is an engineering tool that should be performed early in the design and updated as necessary. It is used to define the high-level airplane or system safety objectives that should be considered in the proposed system architectures. Also, it should be used to assist in determining the DALs for the systems. Many systems may need only a simple review of the system design by the applicant to determine the hazard classification. An FHA requires experienced engineering judgment and early coordination between the applicant and the certification authority.

The FHA is much more than a lengthy derivation; the FHA also includes aircraft and system level requirements that allow the objectives for failure probability to be achieved including:

- *Design constraints*
- *Redundancy considerations*
- *Specification & annunciation of failure conditions*
- *Proposed pilot and crew mitigation action*
- *Recommended maintenance activity*

FHA output data includes:

- *Function list*
- *An FHA worksheet (table) showing the following:*
 - *function identification*
 - *failure conditions*
 - *phase of flight*
 - *effect of the failure condition on the aircraft, flight crew, and occupants*
 - *classification of the failure condition*
 - *verification method that the probability requirement is met*
 - *reference to supporting material*
- *Derived requirements for lower-level systems*

Preliminary Aircraft Safety Assessment (PASA) - New Assessment in 4761A

PASA is a systematic examination of a proposed aircraft architecture to determine how failures could cause the Failure Conditions identified by the Aircraft FHA. The PASA process begins during the initial aircraft architecture development phase. Therefore, early identification of aircraft level safety requirements like function development assurance level, independence and probabilistic budgets, etc. helps to reduce risks during system development process. The PASA is particularly important for evaluating aircraft level failure conditions when multiple systems involved in performing an aircraft function.

Preliminary System Safety Assessment (PSSA)

The PSSA is a set of analyses normally performed during the system requirements and item requirements phases of the aircraft life cycle. The PSSA is where the proposed system architectures are evaluated and defined; this provides the ability to derive system and item safety requirements. The input documents to the PSSA are the aircraft FHA, preliminary aircraft FTA, and the system FHAs. The documents produced during the PSSA are the updated aircraft and system FHAs, updated aircraft and systems FTAs, and the preliminary system Common Cause Analyses (CCAs).

The PSSA identifies derived system safety requirements like redundancy,

partitioning, monitoring, dissimilarity, etc. The PSSA also identifies the necessary Development Assurance Levels for the system functions and items.

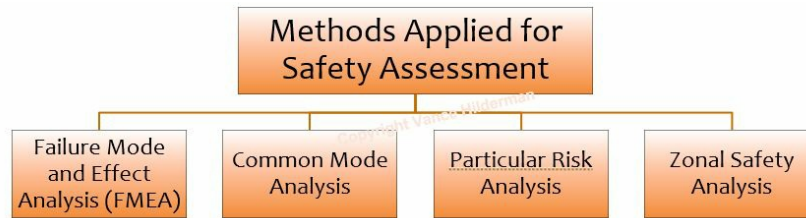
The PSSA is a continuous and iterative process conducted at multiple stages of system development. The objective of the PSSA is to determine how the aircraft and system failures can lead to the hazards identified in the FHAs and to determine how the FHA requirements can be met.

System Safety Assessment (SSA)

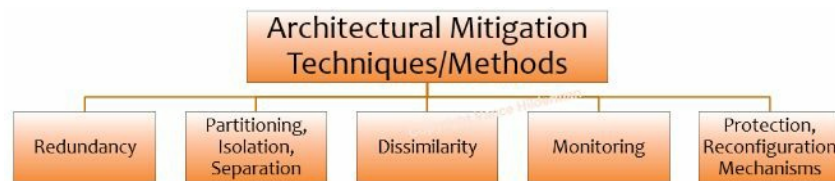
The SSA verifies that the implemented aircraft and system designs meet the requirements of the aircraft/system FHA and the PSSA. The SSA documentation generated during the SSA includes:

- *List of failure conditions from the system FHA and safety requirements allocated to the SSA with FTAs*
- *Results of the qualitative assessments of each failure condition or safety requirement*
- *Any material used to validate the failure condition classifications*
- *Any material shows that assumptions used in the assessment to be valid like flight manual procedures, not-to-exceed intervals, etc.*
- *Documentation showing item installation requirements*
- *If necessary, revised maintenance manuals detailing new maintenance tasks aimed at reducing component exposure times*
- *If necessary, revised flight crew operating manuals detailing procedures to be followed in the event of certain failure conditions*
- *Information showing that the system and items were developed in accordance with assigned development assurance levels.*

The methods applied for the safety assessment are depicted in the following figure:



The most common techniques applied to improve the aircraft and system safety via design enhancements directly supporting improved safety/reliability are depicted in the following figure:



Aircraft Safety Assessment (ASA) - New Assessment in 4761A

Aircraft Safety Assessment is a systematic, comprehensive evaluation of the implemented aircraft to verify that the implemented aircraft design meets the safety requirements as defined in the PASA. The ASA determines that the requirements from the AFHA and PASA have been met. The ASA also demonstrates that aircraft architecture, the relationships between aircraft functions and systems are acceptable.

ASA is kind of final assessment that covers results and evidence of safety assessments performed during development process. The ASA basically covers;

- *The list of Aircraft FHA Failure Condition with the evidence that they are satisfied*
- *Evidence that Safety Program Plan objectives have been achieved*
- *Evidence that aircraft architecture meets the qualitative and quantitative safety requirements*
- *Evidence that aircraft architecture meets the Development Assurance Level allocation requirements*
- *The status of open problem reports and their consequences on the*

aircraft

Fault Tree Analysis

Fault Tree Analysis (FTA) is a top-down analysis technique to determine what single failures or combinations of failures can exist at the lower levels that might cause each failure condition.

The primary purpose of an FTA is to determine the probability of occurrence of the top event -- therefore, demonstrating compliance with a probability requirement specified in a higher level document (usually an FHA). Also, the FTA meets additional objectives:

- *FTAs allow the evaluation of the proposed system architecture enabling the assignment of reliability budgets to systems and items*
- *FTAs identify the need for design modification:*
 - *Added reliability of components redundancy*
 - *Additional redundancy*
 - *Additional monitoring*
 - *Increased maintenance activity*

The FTA Basic Events may get their failure rates from the FMEA.

Failure Modes and Effects Analysis

Failure Modes and Effects Analysis (FMEA) is a systematic, bottom-up analysis performed to identify the failure modes of a system, item, or function and determining the effect of the failure on the next higher level. FMEA's can be done at the component, function, or LRU level. Generally, an FMEA deals with the individual and the combined effects of single failures.

As a minimum, an FMEA should include the following:

- *Identification of the component or function*
- *Failure mode(s) of the component or function*
- *Failure rate for each failure mode*
- *Severity of failure effects*
- *Failure effect at the next higher level*
- *Means of detecting the failure*
- *Compensating actions (i.e., automatic or manual)*

Safety Aspects of Software

Unlike hardware (where you can quantify the safety of an item by calculating its probability of failure over a specified period of time) software safety cannot be numerically measured.

So, rather than specifying probabilities of failure associated with software according to the severity of the failure condition -- the intensity of the software documentation and verification is established according to the severity of the failure condition associated with the failure of the software to perform its intended function.

Software failure condition categories are similar to the hardware failure condition categories described earlier in this white paper. The following table is from ED-12 (same as DO-178):

<i>Failure Condition Category</i>	<i>Description</i>	<i>Software Development Assurance Level</i>
Catastrophic	<i>Failure conditions that would prevent continued safe flight and landing</i>	A
Hazardous/ Severe Major	<i>Failure Conditions that would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be:</i> <i>1. a large reduction in safety margins or functional capabilities,</i> <i>2. physical distress or higher workload such that the flight crew could not be relied upon to perform their tasks accurately or completely, or</i> <i>3. adverse effects on occupants including serious injury or potential fatal injuries to a small number of occupants.</i>	B
Major	<i>Failure conditions that would reduce the capability of the aircraft or the capability of the crew to cope with adverse operating conditions to the extent that there would be, for example:</i> <i>1. a significant reduction in safety margins or functional capabilities,</i> <i>2. a significant increase in crew workload or in conditions impairing crew efficiency, or</i> <i>3. discomfort to occupants, possibly</i>	C

	<i>including injuries.</i>	
Minor	<i>Failure conditions that would not significantly reduce aircraft safety and that would involve crew actions that are well within their capabilities. For example:</i> <i>1. A slight reduction in safety margins or functional capabilities</i> <i>2. A slight increase in crew workload, such as routine flight plan changes, or</i> <i>3. Some inconvenience to occupants.</i>	D

Common Cause Analysis

The acceptance of adequate probability of failure conditions is often derived from the assessment of multiple systems based on the assumption that failures are independent. This independence might not exist in the practical sense, and specific studies are necessary to ensure that independence can either be assured or deemed acceptable. The CCA is concerned with events that could lead to a hazardous or catastrophic failure condition. The CCA is divided into three areas of study:

- **Zonal Safety Analysis (ZSA)** -- The objective of this analysis is to ensure that the system and equipment installations within each zone of the aircraft are at an adequate safety standard regarding design and installation, interference between systems, and maintenance errors.
- **Particular Risks Analysis (PRA)** -- Particular risks are those events or influences outside the systems of interest (for example, fire, leaking fluids, bird strike, HIRF, lightning, etc.). Each risk should be the subject of a specific study to examine and document the simultaneous or cascading effects (or influences) that might violate independence. The objective of the PRA is to ensure that the safety related effects are either eliminated or that the risk is acceptable.
- **Common Mode Analysis (CMA)** -- The CMA is performed to confirm the assumed independence of the events that were considered in combination for a given failure condition. Another way of saying this is that the CMA is performed to verify that combinatorial events in the FTA are truly independent in the actual implementation. The effects of

development, manufacturing, installation, maintenance and crew errors, and failures of system components that defeat the independence should be analyzed.

ARP Certification Software & Platforms

The certification process is complex and time consuming, as we all know. However there are strategies and tools you can apply that will improve the certification process, reducing the resource commitment, time for certification and as a result the cost of certification. Ultimately though these tools can help also improve the safety of your aircraft because less time and resources are committed to performing tasks that can be automated. This can leave more time for engineers to spend on design and safety of the aircraft.

One of the tools often utilized in safety-critical projects is a Model-based tool that creates a Digital Risk Twins (DRT). Known as MADe (Maintenance Aware Design Environment), the MADe platform has added benefits to certification which came through from our analysis. These can be broken into short term and long-term benefits.

Short-term benefits of such tools:

- auto generation of certification documentation,
- standardized taxonomy,
- safety verification PSSA or SSA,
- assisting in the population of FHA, and
- Simulation based analyses.

Longer-term benefits of such tools:

- Knowledge capture mechanism that ensure project knowledge is retained within the organization,
- re-use of retained knowledge assets on other projects,
- ability to adapt design and re-publish required certification documentation easily, and
- ability to close the gap for model-based system design.

Final Words on Aircraft/Avionics Safety

The safety assessment process has fundamental importance in establishing

appropriate safety objectives for the aircraft and systems. The level of safety assessment activities is dependent on the aircraft level failure condition classification and the complexity of the system integration and implementation. Therefore, the safety assessment process should be planned early in the development process and managed through this process.

During the PSSA process while assessing the system architecture, if the FTA shows that the system does not meet the safety requirements (i.e., the calculated probability of the undesired top event is greater than the allowed probability), there are several key upgrades that the system manufacturer can do to improve the situation:

- Use components that have lower failure rates
- Improve Built-In-Test to detect a higher percentage of failures
- Increase the degree of redundancy in the system

Which of the above are best? The answer depends upon cost, availability, and ease of implementation; a trade-off analysis is required to determine the best alternative for your system.

You've now initiated the aircraft and avionics systems safety assessments using ARP4761A, and you're ready to start developing the aircraft and avionics systems themselves. Welcome to ARP4754A (time to turn the page ...).

5

ARP4754A Aircraft & System Development

By Vance Hilderman

Reviewed by:

- Mr. Kevin Crozier, Senior Avionics Software Engineer, FAA Consultant DER, FAA Certified Flight Instructor and Commercial Pilot.
- Mr. Russell McMullan (Former Technical Director - BECA)

More Non-Magic Numbers: 4754, 4761, 79 ...

As shown via ARP4761A for aviation safety assessments, Safety does not rely upon magic numbers but rather real answers. Likewise, Safety is never an accident, but true Safety should prevent accidents. The numbers “4754”, “4761”, and “79” are not magic but are associated with Safety. Safety and the Numbers have evolved – the new answers for Safety are found in 4754A and 4761A; specifically SAE’s ARP4754A (“ED-79” in Europe) and ARP4761A.

The Old Days.

You now know the “Old Days” of technology refer to any time prior to last year. In the Very Old Days (moon landings, space shuttles, commercial jets with four engines) safety was addressed by brains and refinement: smart engineers did their best to prevent accidents then refinement was applied when those best efforts were less than perfect. In the Very Old days, computing horsepower was thinner, schedules were fatter, and acceptable safety generally ensued: space shuttles had a 98.5 percent chance of not exploding and commercial aircraft had fatal crashes “only” a few times annually. Then the very old days gave way to mere old days, and the very old ways didn’t work as well. Some said the engineers were not as smart while others said those engineers were trying to make the computers too smart. But all agreed that more formalized safety was needed and the Society of Automotive Engineers (SAE) had been handling Aerospace Recommended Practices (ARP’s) for

decades. SAE ARP4754, ***Certification Considerations for Highly-Integrated or Complex Aircraft Systems***, was published in November, 1996. Its tightly-coupled sibling ARP4761, ***Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment***, was published the next month, December, 1996.

ARP4754A Today .

The revised ARP4754A is officially titled “***Guidelines for Development of Civil Aircraft And Systems***.” It covers the development cycle for aircraft and avionics systems. Rarely can one judge a book by its cover or title; however, in this case, the title literally conveys a powerful message: if you are involved with development of aircraft or systems, you should be well versed in ARP4754A’s ‘guidelines.’ Why? There are two key points which should be understood before first opening the pages of ARP4754A:

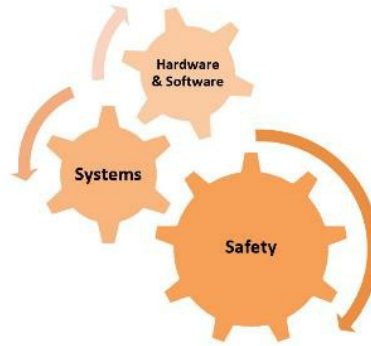
1. ARP4754A’s title states “guidelines,” but failure to understand and apply ARP4754A may reduce safety and will greatly reduce your ability to achieve certification. The ability to demonstrate robust, safe avionics begins with the approach to systems safety before development. It is very difficult to apply retrospectively in order to rectify a weak system.
2. While its predecessor ARP4754 was largely similar, too many organizations treated it as “optional” befitting its name “Guideline”; however, certification organizations worldwide have increasingly, and formally, mandated adherence to this latest version, ARP4754A.

For experienced, proficient developers of aircraft or aircraft systems, ARP4754A reads like a book for maintaining good personal health: make a plan for health, understand healthy living, be safe, eat well, reduce stress, exercise, sleep, get regular check-ups to prove you followed your health plan, and repeat. For aircraft, an analogous synopsis of ARP4754A would state:

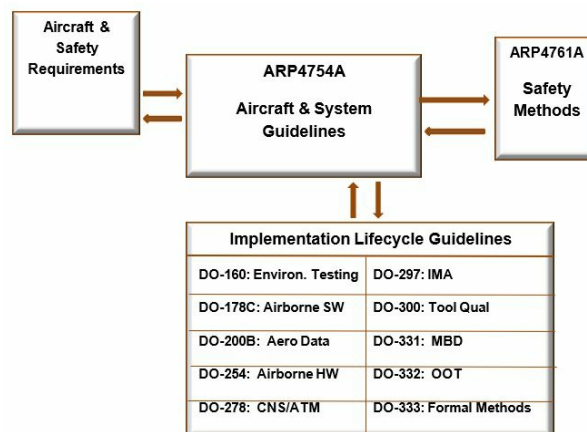
1. Plan your aircraft/system’s development lifecycle ecosystem;
2. Implement Safety activities per ARP4761 (ARP4761A starting in 2018);
3. Define and justify Assurance Level;
4. Define system architecture and requirements; Validate;

5. Perform Verification and Configuration Management;
6. Implement Process Assurance and prove Transition Criteria.

Remember the Safety, Systems & Hardware/Software ecosystem depicted below:



Figuratively and literally, systems development via ARP4754A is the centerpiece: it is preceded by, and must consider, the safety assessment which is used to help define aircraft/system architecture, and aircraft/system safety requirements. In turn, it precedes software and hardware development yet system considerations are continuously addressed during the entire software and hardware development. A refined view of the relevant guidelines is depicted here:



Why ARP4754A? Background .

Before delving into ARP4754A specifics, one should truly consider why it exists. When avionics systems were simpler decades ago, it was possible for smart designers to mentally conceive those systems and proceed immediately

with implementation. Admittedly today, the need for ARP4754A is less justifiable for simple systems. At the same time, the number, variety, and complexity of systems continues to grow exponentially. Clearly, avionics systems can be much more complex than commercial brick and mortar buildings, but it would be inconceivable to begin building a commercial office building without a soil/earthquake analysis, foundation design, and a plan for inspections. Those inspections obviously continue throughout the building process including satisfactory electrical, plumbing, emergency exits, and proper reinforcement. While it is possible great builders could possibly build a safe building without detailed plans, blueprints, processes, and inspections, there would be no way to fully verify the building's "greatness." Why? "Greatness" must be associated with proof a building is great. Proof is based upon assessing implementation versus plans then correcting any deficiencies found.

Clearly, without plans and processes for a building, there is no way to assess, or claim, a building's design and construction are safe. Since developing avionics can be more complex than constructing a building, it is clear that avionics systems require big-picture planning, processes, requirements, safe development, verification/validation, and evidentiary proof of conformance. Welcome to ARP4754A, *"Guidelines for Development of Civil Aircraft And Systems."*

Background: ARP4754

The original ARP4754 standard was first published in 1996 with the purpose of assisting avionics development organizations to think beyond mere hardware and software. Remember, DO-178 (and its European equivalent ED-12) was published over a decade prior to provide guidelines for avionics software. But by the early 90's it was clear that safe software, and software certification itself, required both knowledge of the system and confirmation of system level safety aspects. ARP4754 was focused upon aircraft systems whose failure could potentially affect safety of aircraft or occupants. While there are certainly critical stand-alone components on aircraft which could affect safety, ARP4754 is focused not upon components, but rather systems which have complex interactions with other systems on or off the aircraft. These systems typically involve multiple knowledge domains and are likely

to evolve over time. Thus they are developed by different persons via different disciplines often separated by space and time; the best means to ensure safe implementation is via codified development processes based upon deterministic safety: ARP4754.

The original ARP4754 standard was also written before the onset of more recent avionics development trends such as model-based development (MBD) and integrated modular avionics (IMA). Its focus was upon a top-down, iterative approach to avionics system development where individual system functionality is identified and then increasingly refined. At its inception, the aircraft and systems safety process was less refined and ARP4761 was a mere concept. Also, there was a perception that ARP4754 was an SAE document and therefore not a true dedicated avionics guideline; admittedly, certification authorities typically may not always have required proof that it was being followed. ARP4754 provided a good reference for system development but the aforementioned aspects precluded it from being great. Thus the need for ARP4754A ...

ARP4754A versus ARP4754

ARP4754 in its original form, before being updated to ARP4754A, had the following characteristics:

Old Title: ***“Certification Considerations for Highly-Integrated or Complex Aircraft Systems”***

New Title: ***“Guidelines for Development of Civil Aircraft and Systems***

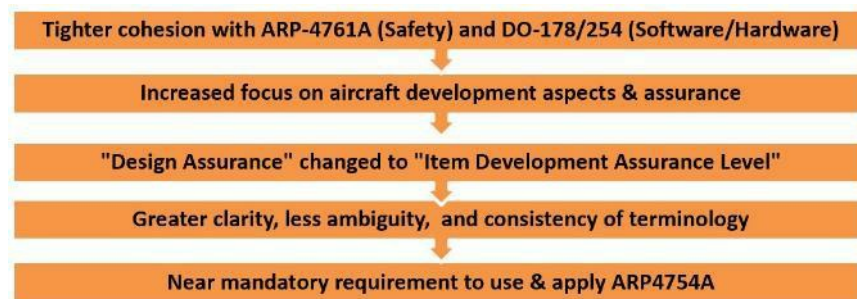
As readily seen, the new title in ARP4754A:

- Emphasizes Development aspects
- Removes ambiguity of “Highly-Integrated or Complex” terminology

Seriously: simply ask “Which aircraft systems on today’s aircraft are neither complex nor highly-integrated”? The answer is “Fewer and fewer ...”. It may seem as a developer of a system that your system is not complex as all the engineers working on the project readily understand it. And each system is physically and electrically separated from other systems so could be considered “not highly integrated”. However, most avionics systems are in fact “integrated” as their safety assessment, design, installation and operation

in fact must consider the aircraft and other systems on that aircraft. Thus ARP4754A's revised and less-subjectively ambiguous title.

With revision "A", e.g. ARP4754A, several key improvements were made as shown below:



ARP4754A Key Attributes & Rationale

ARP4754 was “good”: it described a foundational process for developing safe, good-quality avionics systems and aircraft. However, due to the evolution of related guidelines and certification refinement, and a requirement to address increasing integration and complexity of systems, ARP4754 was considered by many to be incomplete; thus it was not applied as rigorously as needed. The “iterative approach” within ARP4754 was expanded in ARP4754A (in conjunction with ARP4761) to ensure that developers apply a formal iteration process to safety whereby safety is continually addressed with refined safety assessments throughout the project.

By contrast, ARP4754A truly emphasizes the importance of an entire ecosystem for avionics system development, founded upon a formal Safety process (supported by ARP4761). ARP4754A provides specific guidance to complying with regulations, and instructions into a “how to” guide for aircraft and system development, emphasizing the need to integrate that Safety and Systems process continuously throughout development.

On a recent avionics project, this author performed an audit of the system developer's documents, commonly referred to as “artifacts” within the aviation certification regime. The client's Procedures looked more like Plans, and their Audits looked more like Reviews. Clearly it was time to go back to basics. While it would be easy to write chapters on this topic, it is much more challenging to say it all in one sentence. The writer Mark Twain supposedly said something akin to the following in the opening line of a 10-page letter:

“Pardon the length of this letter; I did not have time to make it short.” Meaning: short summaries are more difficult than full novels. We hate long novels in avionics: too much fluff and serious developers lack the time for that. Here is an all-too-brief summary of Plans, Procedures, Reviews, and Audits for avionics systems:

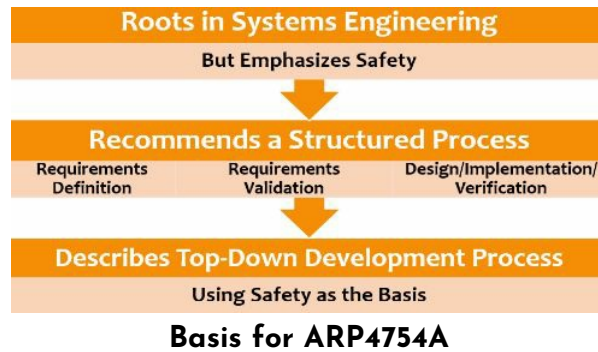
- “Plans” comply with safety requirements and summarize what you will do, while
- “Processes” state how you will implement the Plans, and
- “Checklists” denote objective review criteria to determine if Processes were followed, then
- “Process Assurance Audits” assess conformance of engineering/manufacturing activities, including respective transitions to those Processes and Reviews.

Now, who are the intended users of ARP4754A? The primary ARP4754A stakeholders are summarized in the following graphic:



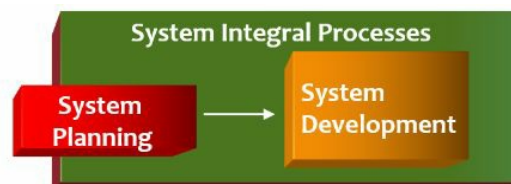
Primary ARP4754A Stakeholders

Remember the parable about the different blind persons touching an elephant? Many ARP4754A novices likewise have different perspectives based upon their own vision: an aircraft safety and design guide, or a system development guide, or a “how-to” manual to integrate ARP4761A’s safety assessment regimens. But which is it? ARP4754A is all those things but in a generic fitness-manual format based upon provable science. This is summarized in the figure below:



ARP4754A's Three Key Processes

In an oversimplified form, ARP4754A requires a System Planning process to be performed covering eight planning topics described below. After these eight ARP4754A planning topics are addressed within approved plans, the System Development process begins; system development does not include actual hardware or software development within ARP4754A since those are covered under DO-254 and DO-178C respectively. In the background, the system Integral Processes are continuously performed. The relationship depicted between these processes is depicted in the figure below; note that the arrow between System Planning and System Development is intentionally drawn one-way, not bi-directional: once Plans are approved, there is no premeditated intention to update those already-approved Plans. (If later there are deviations to the Plans, the deviations are approved and documented externally to the Plans.)



ARP4754A's Planning Process

What do aircraft, complex systems, and simple systems alike have in common? They come under the purview of ARP4754A. It is almost impossible for one guideline to be all things to all people; ARP4754A strives to come close. Since aircraft and systems can have huge variations, ARP4754A avoids mandating a prescriptive approach. Instead, guidance is provided for the developer to consider the ramifications of safety and functionality external and internal to their scope of development and proceed

accordingly. The analogy is that of a personal fitness coach whose goals vary dramatically depending upon their coaching an injured patient, an average office-worker, or an Olympic athlete: different problem domains require a different focus. But there is only one ARP4754A so the guidelines must be generic enough to satisfy different domains. Central to any aircraft/system development process is ARP4754A's required Planning process.

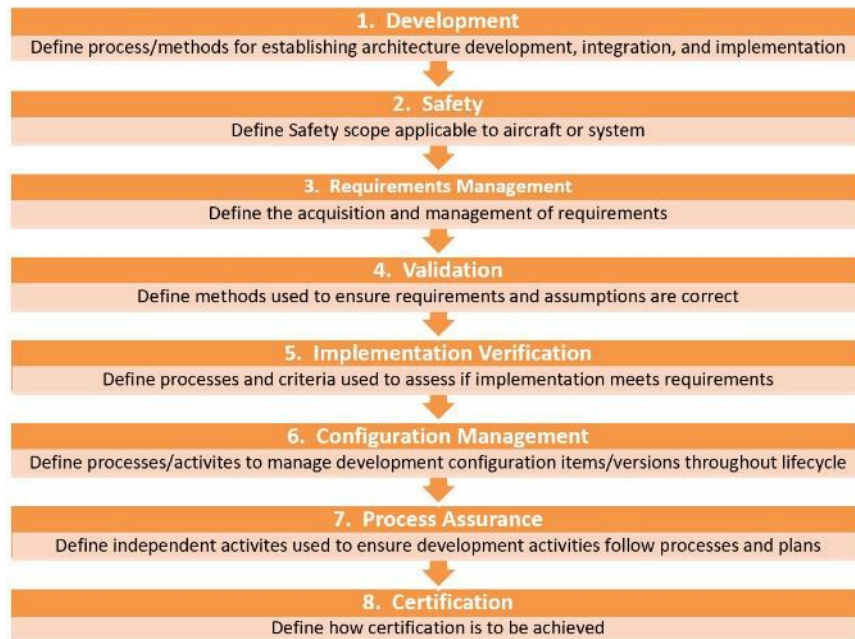
ARP4754A provides a comprehensive guideline for performing the myriad engineering activities necessary to develop safe, high-quality avionics systems. As all pilots know, it's both common and helpful to file flight plans prior to takeoff for significant flying activity. Such flights plans fulfill three primary purposes:



ARP4754A's planning process is thus similar to a flight plan as it has three primary purposes:



Once defined, can the system development plans change? Certainly! Just like flight plans can change, the real life of avionics development is rife with small changes. However, changes to the ARP4754A planning process need to carefully consider overall safety aspects, both within that system and also to other systems, meaning both air and ground based. And they must be documented and approved. The ARP4754A Planning process requires consideration and documenting the following key development aspects depicted below:



ARP4754A's Eight Planning Topics

How the above planning details are placed within documents is less important than the planning details. It is common to make separate plans for each of the above aspects (though Verification and Validation are often combined); separate planning documents increase their reusability on future projects. The question always arises: “How detailed should the plans be?” The answer of course depends upon the complexity of the associated engineering activities. Planning to build a sky scraper requires more details than planning to build a small house. Same for aircraft and systems. Plans should be more detailed in areas of the system identified as offering risk, but acknowledge that you may not know these until you’re underway. Appendix A of the ARP outlines the process objective requirements – (independence required, recommended, as negotiated, and not required). There should be sufficient detail to deterministically answer the following two questions:

1. Can a certification authority review the plans and make a determination that the defined activities could lead to provably safe development?
2. Can detailed checklists for each planning activity be made by extracting information from the plans for the purpose of independently assessing corresponding development activities?

Planning Elements	Element Description	4754A Section
Development	Establish the process and methods to be used to provide the framework for the aircraft/system architecture development, integration and implementation.	This section and 4.0
Safety Program	Establish scope and content of the safety activities related to the development of the aircraft or system.	5.1.5
Requirements Management	Identify and describe how the requirements are captured and managed. Sometimes these elements are included in conjunction with the validation elements.	5.3
Validation	Describe how the requirements and assumptions will be shown to be complete and correct.	5.4
Implementation Verification	Define the processes and criteria to be applied when showing how the implementation satisfies its requirements.	5.5
Configuration Management	Describe the key development related configuration items and how they will be managed.	5.6
Process Assurance	Describe the means to assure the practices and procedures to be applied during system development are followed.	5.7
Certification	Describe the process and methods that will be used to achieve certification.	5.8

Scope of ARP4754A's Planning Process

Quick Note on Validation, Verification, Process Assurance, Reviews & Audits

Verification is performed by engineering and assess es if implementation meets requirements. Validation assesses the acceptability of those requirements. In the avionics world (and reading between the lines of ARP4754A and the DO-XXX documents) it should be understood that at its simplest, “Verification = Reviews + Tests + Analysis.” Virtually everything is reviewed. Requirements and implementation are tested. Analysis is applied when reviews/tests are not 100 percent conclusive. Both validation and verification are addressed later herein.

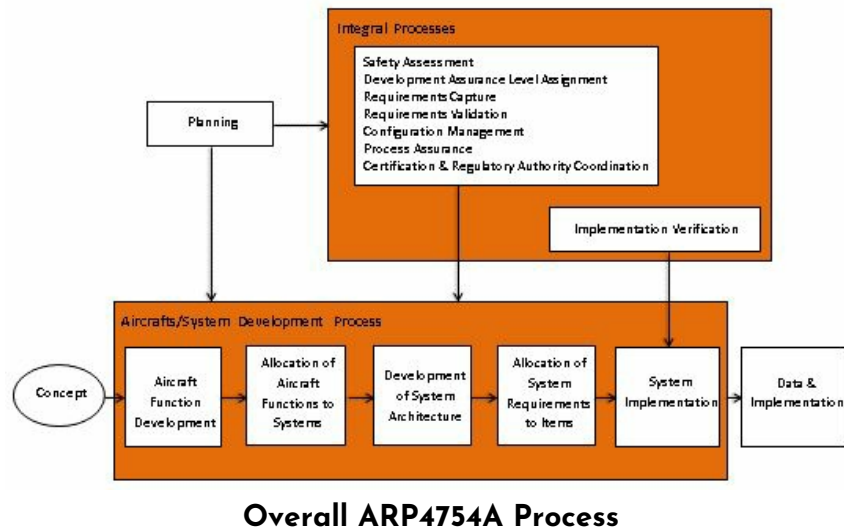
Audits are process-oriented and performed by independent Quality Assurance (QA), also called Process Assurance (PA), personnel. They are not reviews; instead audits determine if Engineers followed defined processes. For Systems, quality assurance is termed Process Assurance (PA) which is a superset of quality assurance, but the terms are used in aviation almost interchangeably.

A major problem happens when Reviews are incomplete because an engineer thinks an auditor will review later; they won't. Auditors do not do reviews; period. They do audits. Reviews are technical, audits are to process. Both are needed: each is important but they are different. Like a wine bottle and the wine itself: cannot enjoy one without the other - which is more important? Both ...

Another problem occurs when Auditors think they are Reviewers. They are

not; if an Auditor (Process Assurance) has to review, then the question is “Why didn't Engineering do an appropriate review, as documented by the corresponding Checklist?”

Overall, the ARP4754A process is depicted below:



Safety First

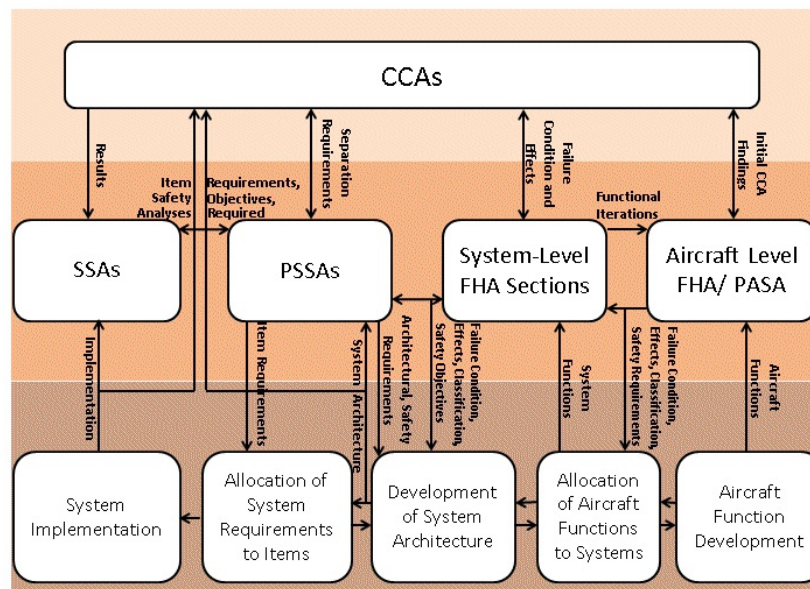
You have planned your avionics system development according to the above and the plans are complete. Where do you start? Easy: with the Safety process. Safety truly is the foundation and primary focus of ARP4754A: the reason for all the plans, processes, and checklists is to enable Safety to be kept at the forefront of development and assessed throughout.

At the aircraft level, ARP4754A requires an overall Safety Program Plan (SPP). It is common to have system-level SPP's for each system. Regardless, each + is considered within the aircraft level SPP. Like building a house, the scope of avionics safety takes many different forms: foundation, architecture, problem detection, problem mitigation, susceptibility to particular hazards (lightning, etc.), and verifiability. So multiple safety activities and analysis are required. The following figure summarizing the primary safety activities is repeated below:



Primary Avionics Safety Activities

For a more complete description of Functional Hazard Assessment (FHA), PSSA, SSA and CCA, see the corresponding ARP4761 Safety Assessment chapter herein. A summary of the safety processes is depicted below:



Safety Assessment Process

Functional Hazard Assessment

The objective of the FHA is to identify functions, failure conditions of functions (loss of function, malfunction, etc.), and their classification (catastrophic, hazardous, etc.) so that aircraft and system designs may be proposed and achieved which decrease the probability of the occurrence of the failure conditions to acceptably lesser levels. It should be noted that in some cases, total loss of a function is not as impactful as partial or misleading

loss of functionality; a pilot may be more likely to mitigate total loss of one system's functionality by switching to a different system.

Preliminary System Safety Assessment (PSSA)

The PSSA is a set of analyses normally performed during the system requirements and item requirements phases of the aircraft life cycle. The PSSA is where the proposed aircraft and system architectures are evaluated and defined; this provides the ability to derive system and item safety requirements. The input documents to the PSSA are the aircraft FHA, preliminary aircraft Fault Tree Analysis FTA (and allocation of failure budget against the top level failure requirement), and the system FHAs. The documents produced during the PSSA are the updated aircraft and system FHAs, updated aircraft and systems FTAs, and the preliminary system Common Cause Analyses (CCAs). The PSSA is a continuous and iterative process. The objective of the PSSA is to determine how the aircraft and system failures can lead to the hazards identified in the FHAs and to determine how the FHA requirements can be met. The PSSA is often associated with the Preliminary Aircraft Safety Assessment (PASA). An important note: Appendix 1 of AC23.1309-E has some useful guidance to assist with assessing the severity of failure conditions of aircraft functions and systems at this stage of the project.

System Safety Assessment (SSA)

The ARP4754A / ARP4761A SSA verifies that the implemented aircraft and system designs meet the requirements of the aircraft/system FHA and the Preliminary Aircraft Safety Assessment (PASA) and Preliminary System Safety Assessment (PSSA). The SSA documentation generated during the SSA includes:

- *Updated aircraft FTAs, system FHAs, and system FTAs*
- *Documentation showing item installation requirements*
- *Any material used to validate the failure condition classifications*
- *If necessary, revised maintenance manuals detailing new maintenance tasks aimed at reducing component exposure times*

- *If necessary, revised flight crew operating manuals detailing procedures to be followed in the event of certain failure conditions*

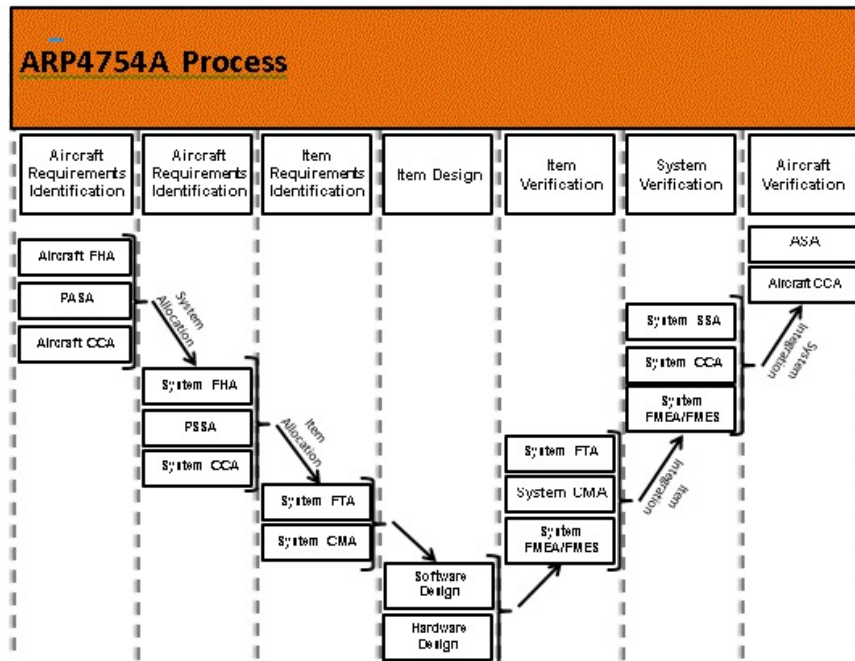
The SSA should verify that all significant effects identified in the FMES are considered for inclusion as primary events in the FTA, and the SSA should also include applicable CCA results.

Common Cause Analysis (CCA)

The acceptance of adequate probability of failure conditions is often derived from the assessment of multiple systems based on the assumption that failures are independent. This independence might not exist in the practical sense, and specific studies are necessary to ensure that independence can either be assured or deemed acceptable. The CCA is concerned with events that could lead to a hazardous or catastrophic failure condition. The same software installed on multiple LRU's is considered by many to be the most often overlooked area of common cause. For example, Dual Flight Management Systems (FMS) could fail under the exact same conditions if the same software is installed on both units.

For a more complete description of FHA, PSSA, SSA and CCA, see the corresponding ARP4761 Safety Assessment chapter herein.

The overall Safety process relationship within ARP4754A is depicted below:



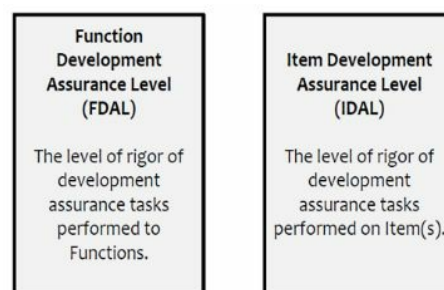
Development Assurance Level Assignment

In conjunction with the Safety activities above, the Development Assurance Level (DAL) assignment is determined. Informally referred to as “criticality level”, the DAL is a designator which defines the development process rigor associated with the corresponding probability that errors could occur. The more critical the item to flight safety, the less the probability that errors can occur for that item.

There are two types of DALs defined by ARP4754A:

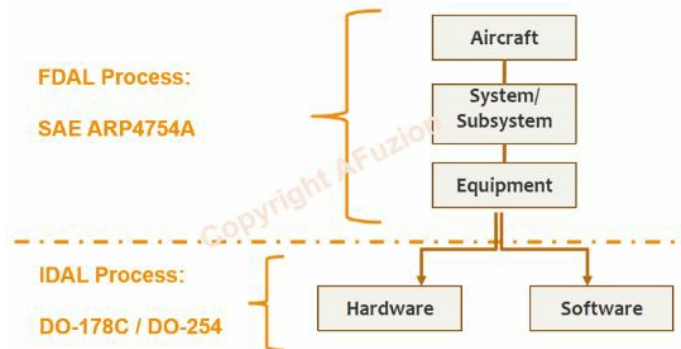
1. Functional DAL, called FDAL; and
2. Item DAL, called IDAL.

FDAL and IDAL are summarized in the figure below:



FDAL Versus IDAL Summary

The following figure depicts the different allocations between FDAL and IDAL:



FDAL & IDAL Aircraft, System, and Item Perspective

Remember: the safety process is iterative, however initiation of FDAL assignment should precede IDAL assignment; if you don't know the FDAL, you cannot fully assign the ensuing IDAL item Objectives per FDAL:



The Two Design Assurance Level Types: FDAL & IDAL

While not mandatory, it's most common to determine the FDAL first, then followed with IDAL determination. Why? Functions are associated with requirements, and those requirements are allocated to items. Functions and their requirements thus describe “what” a system does while the actual requirement is implemented in hardware and/or software items. ; This is to allow a single function to be split between multiple items to potentially allow a lower Design Assurance Level of those individual items. The level of rigor for the functional requirement development is defined by the FDAL, with level A being the most rigorous. The required development objectives for each FDAL are provided in Appendix A of ARP4754A itself. The corresponding hardware/software item development likewise has a specific

DAL assignment called an IDAL; the required hardware/software development objectives for each IDAL are provided in DO-178C (ED-12C for Europe) and DO-254 (ED-80 for Europe).

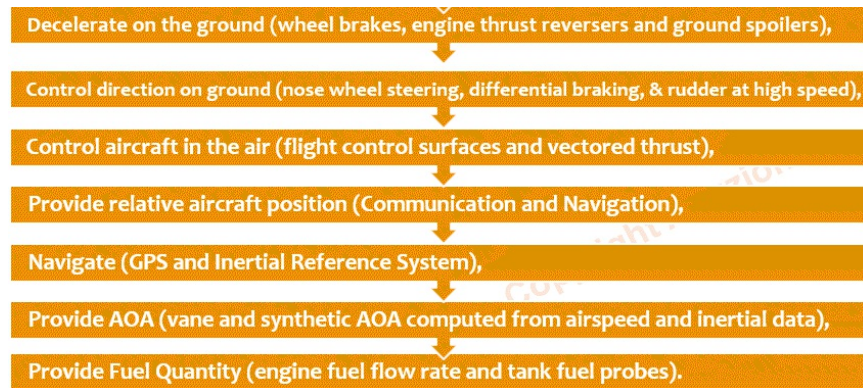
The DAL determination starts by considering the functions in the aircraft or system's FHA failure conditions; therefore these are called "Top-Level Failure Conditions." The FDAL is assigned based upon the most severe Top-Level Failure Condition. For example, if the most severe is DAL B, the FDAL is thus Level B. A synopsis of the FDAL's is provided below:

Top-Level Failure Condition Severity Classification:	Corresponding FDAL Assignment
Catastrophic	A
Hazardous/Severe-Major	B
Major	C
Minor	D
No Safety Effect	E

FDAL Assignment Allocations

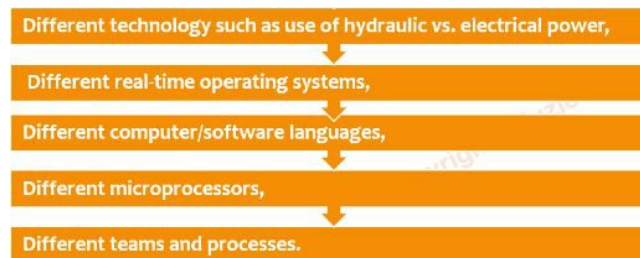
ARP4754A describes specific processes for considering independence between systems and processes as they pertain to DAL. The goal is to ensure requisite levels of necessary safety and elimination of common mode failures. For example, if a single failure can affect multiple systems or components, then independence can be added to prevent that single failure from affecting more than one system or component.

Functional independence ensures that functions are not affected by common errors within the same function. For example, navigation is clearly an important function for aircraft safety; using two sources of navigation data (Global Positioning System (GPS) and Inertial Navigation System (INS)) provides for functional independence and elimination of many common mode error problems which could occur otherwise:



Examples: Functional Independence

Item Development Independence likewise ensures that the items associated with the function do not experience common errors; for example using different operating systems within independent items provides for Item Development Independence which could occur if a single operating system had an unmitigated error:



Examples: Item Independence

Defining Requirements

You have the DAL's defined and it is time to define requirements. What? Requirements are not considered during the DAL assignment process? While it's common to know, or at least initiate, the requirements in parallel with the DAL assignment process, the DAL is about safety and thus should be initiated without formally considering the requirements. Safety considerations need to precede functionality. Requirements are defined partly to satisfy safety, as determined via the DAL process. So DAL determination precedes requirement definition. How are requirements defined? By considering the various types of requirements and how each pertains to the aircraft or system being developed. The following figure depicts the primary types of requirements which must be defined, along with the common (but not mandated) order:



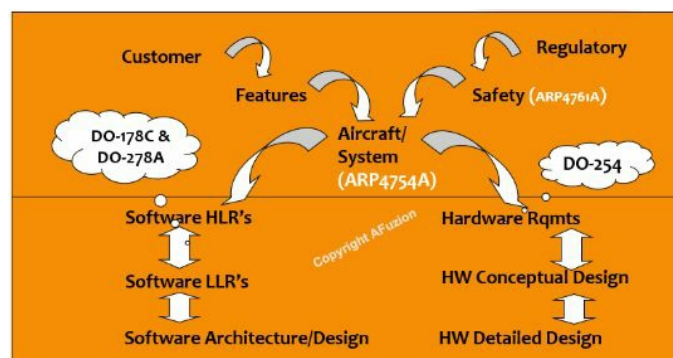
Primary Aircraft/System Requirement Types

Which of the above requirement types are important? They all are. But really, which are most important? Well, consider: functional and operational performance are important: the aircraft and systems need to fulfill objectives and those objectives must be clearly defined; requirements provide that definition. Operational aspects and the interfaces which effect the ability to perform and necessary to define. However, safety is paramount, so the safety requirements must be paramount. But safety requirements are usually the most difficult to define as there is more subjectivity involved: how safe is “safe”? Careful consideration must be given to the safety assessment artifacts to consider relevant failure conditions, modes, and effects; knowledge should focus upon prevention but wisdom should prevail knowing one hundred percent prevention is impossible. So architectural aspects including redundancy, dissimilarity, partitioning, etc. must be added to support the requisite DAL, and those aspects must be defined via requirements. The approach to be used for requirements capture must be defined in advance, in the planning process. Why? Plans are independently assessed prior to following the activities described in those plans, so the planned requirements capture process can be evaluated prior to initiation. Some say the biggest mistake made during this phase is for the requirements writers to incorrectly muse “but the pilot would never do that ... “ Particularly as flight decks become more and more automated, the Catch-22 of automation pervades: the more automation, the more the pilot depends upon automation, so the more

automation ...

Requirement Sources and Traceability

The aviation development ecosystem places “requirements” as the foundation for success. When Fred Brooks wrote “The Mythical Man Month” in 1975, none of these aircraft, system, safety, hardware, or software guidelines were in existence; they came just a few years later. But Brooks’ monumental book on complex computing systems reached important conclusions later “baked” into the emerging aviation guidelines. Brooks noted the number one source of errors for such computing systems was “assumptions” including assumptions made by engineers in the absence of defined requirements. Remember: “Verification” is the assessment of implementation conformance to requirements, whereas “Validation” is the assessment of the requirements themselves. Validation is more important because if you don’t correct, complete, and unambiguous requirements, it doesn’t matter later if you in fact “verified” them. (This is all covered in more detail below and in subsequent chapters). So ARP4754A mandates a requirements management process and such is flowed down through the hardware and software processes as depicted in the following figure:



Aviation Ecosystem Requirement Sources & Relationships

Regarding safety and requirements, an often misunderstood aspect is the role of derived requirements. What are derived requirements? Be careful of linguistics. If English is not your native tongue, despair not: native English speakers are seemingly less able to understand the meaning of “derived requirements” because they are derived from an engineer’s desire to address an otherwise undocumented or overlooked capability. Derived requirements usually do not trace to a parent requirement or describe obvious functionality.

Instead, derived requirements are additional aspects which the system or aircraft being developed must implement, typically to address safety, re-use, or verifiability. For example, sampling rates, use of partitioning, or adding redundancy with health monitoring and voting are all examples of functionality commonly addressed via derived requirements. Since derived requirements often relate directly to safety or to a gap-closing definition of functionality, ARP4754A requires developers to ensure the existence of a feedback loop to the safety process for all derived requirements. Any change to derived requirements must be fed back to the safety process to determine any direct or indirect impact upon safety and the developers are required to prove the mandatory use of this process for all derived requirements, as audited by QA.

Requirement Validation

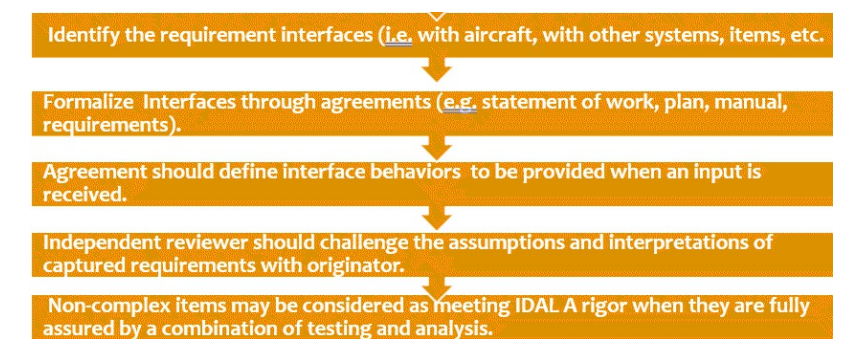
Congratulations – the aircraft and system requirements are captured and it's time to initiate design then implementation. But you're smart and you know you will verify those requirements later to ensure implementation satisfies the requirements. Wait: how do you know you have the "right" requirements? Exactly. That is the role of requirements validation and it must precede design/implementation. Why? Requirements are the foundation to aircraft and systems. Like building a house, changing the foundation after starting construction is undesirable for many reasons, most important of which is safety. Truly, you need to determine if you have conflicting, incomplete, or incorrect requirements long before you initiate development. ARP4754A requires a formal validation process whereby requirements are evaluated for correctness and completeness before they are used in design/implementation. However, requirements validation is performed throughout the development process because changes are inevitable and all changes to requirements or anything affecting a requirement must be assessed through validation. The approach to requirements validation must be defined in advance via plans; typically a separate validation plan is utilized, however it is common to combine validation planning documentation within the verification plan since validation and verification are related.

Remember: Aircraft, System, and Hardware requirements can and must be Validated. However, since "Validation" includes "completeness and

correctness”, software requirements (discussed later in this book) are NOT validated since software requires both hardware and a system context to be assessed for Completeness.

Requirements validation is a formal process, and within aviation, most formal processes require checklists. Even a 5,000 hour pilot with vast experience is formally required to follow a checklist for many key activities including takeoff and landing. And because safety is so important, commercial aircraft utilize a co-pilot for added independent verification. Fortunately, ARP4754A provides high-level requirement validation criteria in section 5.4.3 which should be used to form the basis of the validation checklist. For brevity, these are not repeated herein as there is little unique about requirements validation in aviation versus other safety-critical industries (however the term “validation” itself is used differently – see the corresponding Aviation Requirements chapter in this book). The usual evaluation criteria apply: correctness, non-ambiguity, avoidance of conflicting logic, identifiability, traceability, impact upon safety, etc.

The following figure provides helpful notes for Aircraft and System requirements validation:



Aircraft & System Requirements Validation: Helpful Notes

The amount of validation rigor varies according to the criticality, e.g. “DAL” of course. The Table below reprinted from SAE’s ARP4754A clearly explains. Note that “R” in the Table supposedly means ‘Recommended’ as stated; however, in reality most worldwide certification authorities lend a more conservative interpretation to instead imply “Required”. It’s this author’s experience that certification authorities almost always Require the Recommended validation methods.

Methods and Data (see 5.4.6.a-f and 5.4.7)	Development Assurance Level - A and B	Development Assurance Level - C	Development Assurance Level - D	Development Assurance Level - E
PASA/PSSA	R	R	A	N
Validation Plan	R	R	A	N
Validation Matrix	R	R	A	N
Validation Summary	R	R	A	N
Requirements Traceability (Non-Derived Requirements)	R	R	A	N
Requirements Rationale (Derived Requirements)	R	R	A	N
Analysis, Modeling, or Test	R	One recommended	A	N
Similarity (Service Experience)	A		A	N
Engineering Review	R		A	N

Note: R - Recommended for certification, A - As negotiated for certification, N - Not required for certification

For each requirement, a combination of the recommended and allowable methods necessary to establish the required confidence in the validation of that requirement, should be identified and then applied.

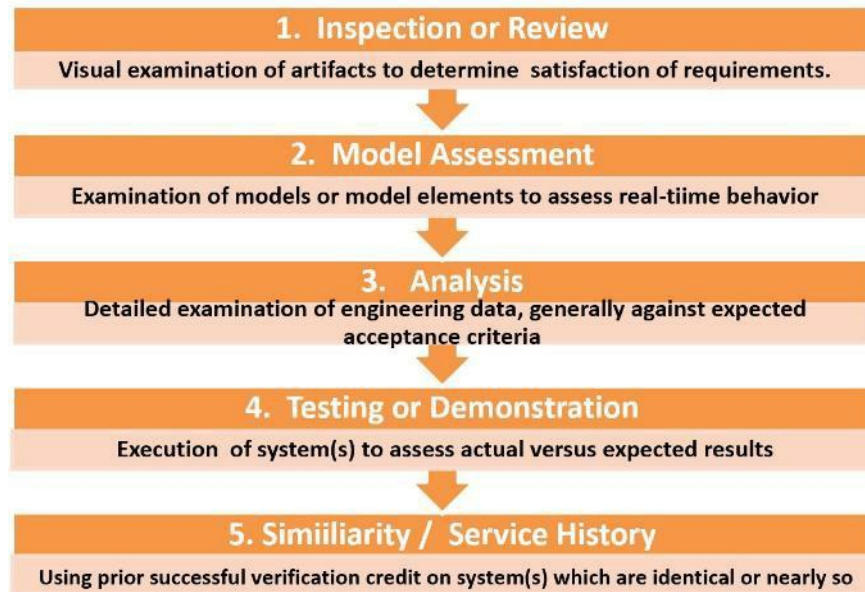
Implementation Verification

The plans have been followed perfectly, requirements seem to be perfectly validated, and QA has audited the activities throughout the engineering process including all transitions through implementation. Ready to begin flight test? Not so fast. Remember, in aviation, the probability of software error is assumed to be “1”, e.g. errors happen – can they be detected and mitigated? During engineering, the same philosophy applies. While great plans, processes, and engineering produce measurably better systems with fewer errors, verification is still required. The common belief is that verification improves quality by finding errors. While not incorrect, verification is really performed to assess the adequacy of prior activities in conjunction with a feedback loop to improve those activities and the implementation. Therefore aviation verification is more important, and more encompassing, than mere bug detection.

It is commonly believed that verification is performed purely to identify errors so that they can be fixed. While partially true, verification at the aircraft and avionics systems level per ARP4754A is more encompassing, as it is also used to assess the completeness and quality of preceding development activities including Safety, Requirements, and Design; not just Implementation. Remember the overly simplistic verification equation cited earlier,

$$\text{“Verification = Reviews + Tests + Analysis”}$$

Both the required, and the potential, verification activities are more diverse; keep in mind the degree of verification rigor increases with the DAL rigor, e.g. DAL A is the most intensive. The following figure depicts the common types of verification applied under ARP4754A:



Like Validation described in the preceding section, Verification rigor varies according to DAL. This is fully described in the following table reprinted from SAE ARP4754A:

Methods & Data	Development Assurance Level				
	A	B	C	D	E
Verification Matrix	R	R	R	A	N
Verification Plan	R	R	R	A	N
Verification Procedures	R	R	R	A	N
Verification Summary	R	R	R	A	N
ASA / SSA	R	R	R	A	N
Test	R	R	R (1+)	A	N
Inspection, Review or Analysis	R (1+)	R (1+)		A	N
Test – unintended function	R	R	A	A	N
Service Experience	A	A	A	A	A

R – Recommended for certification, A – As negotiated for certification
N – Not required for certification

Aircraft/System Supporting Processes per ARP4754A:

Configuration Management & Process Assurance

Your engineers are smarter than Einstein and more organized than a

contingent of efficiency experts; you have built the world's first perfect aircraft system on the first attempt. Kudos. Surely certification will be a breeze, correct? Hardly. Perhaps your engineers were utterly brilliant but what about the weaker link who is hired later to make changes to the system; that is why professional configuration management (CM) is needed. And how can you prove your flock of Einstein's actually followed the defined process? Independent process assurance (PA) must audit the engineer's work and artifacts to assess conformance to defined processes. CM and PA are supporting processes whose output does not directly contribute to the aircraft/system, but which are completely necessary.

Configuration Management (CM). CM must be defined within a plan and then carried out according to that plan throughout the life of the aircraft/system; not just through initial certification. Essentially, CM ensures the ability to know, and control, the provenance of all elements comprising every deployed aircraft/system and corresponding artifacts. Proper CM is in place when everyone could leave a project and subsequent personnel would be able to perfectly recreate any delivered aircraft or system version and know the full history of all contributors, dates, versions, and relationship to other versions, with assurance that no unauthorized changes have been made to any such version. Therefore, CM processes must be in place to establish baselines, manage change control, ensure archival and retrieval, and maintain a perfect configuration index which lists every artifact used in the engineering of the system/aircraft. Always remember: the more complex the project or the more volatile the requirements, the greater the probability to introduce configuration management which are nearly impossible to recover from without excessive undue effort.

Process Assurance (PA). PA was lightly touched upon earlier above when audits versus reviews were addressed. Remember: engineers perform reviews which are technical; process assurance performs audits which are process-oriented. Again, within PA, the commercial and conventional military world are different than ARP4754A and DO-XXX. PA has different roles within the avionics development ecosystem: PA approves the plans, standards, and checklists then assesses compliance with such; they don't do reviews - they do audits including assessment of transition criteria. Avionics development plans specify major activities in terms of inputs and outputs for each of those

activities. Those inputs and outputs must be adhered to by engineering and assessed via reviews. Reviews are done by engineering, using engineering checklists. PA assesses those inputs and outputs and checks as ‘transitions’, and thus determines if the transition criteria were met. Audits are done by PA, using PA checklists.

CM and PA processes must be defined in a plan. CM need not be independent, meaning engineers can perform their own CM though large organizations tend to have CM teams or individuals solely responsible for such. PA however must be provably independent from engineering, and PA members should not report to the system/aircraft engineering manager but rather through a separate and independent quality assurance organization in order to provide demonstrable independence.

The overall system processes of ARP4754A have now been described. Additional guidelines discussed in subsequent chapters herein exist for airborne software and hardware, databases, ground based systems, and virtually every other subset of aircraft and systems which related to engineering quality or safety. Each of those entities has specific guidelines dedicated to their corresponding area but which largely mimic the foundational processes specified within ARP4754A: safety, planning, requirements, implementation, V&V, CM, and QA. Build a good aircraft and system foundation, and the remainder of the building can be solid. But remember: anyone can overpay for custom solutions and with excess time/schedule achieve a satisfactory result. The wise achieve better results, more quickly and more cost-effectively, by knowing their goals early and planning accordingly.

6

DO-178C Avionics Software Prelude

By Vance Hilderman

Reviewed by:

- Ms. Gamze Eroglu, Former Senior Certification Specialist, STM
- Mr. George Meier, FAA DER
- Mr. Bill Potter, Technical Marketing Manager, The MathWorks

By 2005, DO-178B, the avionics software guideline titled “*Software Considerations in Airborne Systems and Equipment Certification*”, was becoming middle-aged and it was simply time for a refresh. What could be easier? Admittedly, the world of software development, even within the usually staid avionics community, had made many strides which DO-178B published fifteen years prior could not possibly have foreseen. In technology as in life, the only constant is change. Especially software technology where change is measured in months, not decades.

As described in the preceding chapters, DO-178 (or its identical counterpart “ED-12” in Europe, where “ED” simply stands for “European Document”) for avionics software is part of the overall aviation development and certification ecosystem. In fact, avionics software development theoretically and normally follows the other key processes of 1) aircraft/systems safety, 2) aircraft and systems development, and 3) hardware development. But the real world is often different from the theoretical, and such is the case here: the first version of DO-178 was published in 1980, which was LONG before the other guidelines for Safety, Aircraft/Systems, and Hardware. Therefore, instead of being a simple cog in the wheel of aviation, DO-178 is in fact the historical and foundational cornerstone for the other guidelines and standards. It should come as no surprise then that the myriad aviation guidelines which were published after DO-178 all look familiar: it is because they deliberately mimic DO-178 which actually promotes consistency.

In human and universal history, specific dates are rarely important

themselves; it's rather the surrounding context which bears import. Same with DO-178: remember, we designed the systems which put mankind on the moon in the mid-1960's, without any DO-178. However, those rockets and lunar systems had relatively small software codebases developed by superbly smart engineers without regard for extensibility. Apollo was also not as concerned with average/weak pilot skills, untrained passengers, weather extremes, traffic avoidance, etc. However, by the mid-1970's the rapid increase in computerization developmental dispersion of civil aviation mandated a verifiable framework of consistent avionics software engineering processes. The number "DO-177" was already taken by the 177th document in the "DO-XXX" series so the next available number of "178" was allocated to this nascent "guideline". An all-too-brief history of DO-178 is shown below:

Doc	Year	Basis	Themes
DO-178	1980 - 1982	Nascent SW Frameworks	Artifacts, documents, traceability, testing
DO-178A	1985	DO-178	Processes, testing, components, four criticality levels, reviews, waterfall methodology
DO-178B	1992	DO-178A	Integration, transition criteria, diverse development methods, data (not documents), tools
DO-178C	2012	DO-178B	Reducing subjectivity; Address modeling, detailed requirements, OOT, Formal Methods: "Ecosystem"

Overly Simplified DO-178 Evolution History & Context

Remember the historical context: by the late 1970's when the need for avionics software guidelines became necessary, true software "engineering" was in its infancy. Both "Computer Science" (coding) and "Computer Engineering" (hardware development) were maturing, but the "process" of developing large-scale or complex software was still informal. Brook's book "The Mythical Man Month" summarized the challenges of such software development yielding answers such as improved software requirements, and communications, to minimize assumptions. Thus was the birth of DO-178. Rather like a successful country's Constitution, guidelines for a rapidly evolving field such as software and aviation are best served by emphasizing flexibility over prescriptive recipes. Thus DO-178 did not, and still does not, address "how" to meet the Objectives. Instead, DO-178 describes the Objectives and how an increasing number of them must be met as the software criticality (termed Development Assurance

Level - DAL) increases until reaching the maximum at 71 Objectives for DAL A software in DO-178C today. The authors of DO-178 were primarily comprised of engineers employed by the world's largest avionics designers/manufacturers. While Boeing and Airbus (and many other airframers) design and integrate aircraft, those airframers are not heavily involved in software development. And unlike other safety-related industry standards largely developed by independent standards bodies or government personnel, DO-178 is unique in being developed by commercial engineers representing the very companies which would uphold DO-178.

Fast Forward 25 Years ...

Because of the continued evolution in software development tools and methods, there needed to be updates to avionics certification strategies. These could have been addressed via FAA Advisory Circulars ("AC's") or even Certification Authorities Software Team (CAST) position papers. AC's and CAST papers were continually being published to accomplish just that. Admittedly, it would have been imminently quicker and simpler to issue augmentation papers instead of updating DO-178B itself. Basically, the avionics development community was faced with making one of four choices for future avionics development and certification strategy:

1. Do nothing.
2. Give DO-178B a minor tune-up.
3. Give DO-178B a major tune-up
4. Overhaul DO-178B.

Clearly, "doing nothing" was not a viable option as both technological advancements and savvy developers (some of whom increasingly gamed the system to "leverage" shortcuts) meant DO-178B was outdated. And a total overhaul of DO-178B was unnecessary: unlike the major overhaul from DO-178A to DO-178B, the vast majority (over 98 %) of DO-178B was deemed "good" so an overhaul was not needed. That leaves option "2" and option "3" above to choose from. Depending upon your perspective, the DO-178B DO-178C update which literally changed 2% of DO-178B was either "minor", or "major": if considering the quite modest changes to DO-178B itself, the DO-178C updates are very minor. If instead considering the Ecosystem of the four new supplements added pertinent to the DO-178C and related aviation guidance

ecosystem, the inclusive updates were quite major. Those supplements are summarized later in this chapter and dealt with in separate chapters within this book.

There were two issues that needed to be addressed to breathe new viability into DO-178B:

1. Provide guidance for acceptable use of new software engineering technologies including advanced Software Tools, Modeling, Object Oriented Technology (OOT), and Formal Methods (FM).
2. Minimize the ability of savvy developers to take shortcuts while they purported to “comply” with DO-178B.

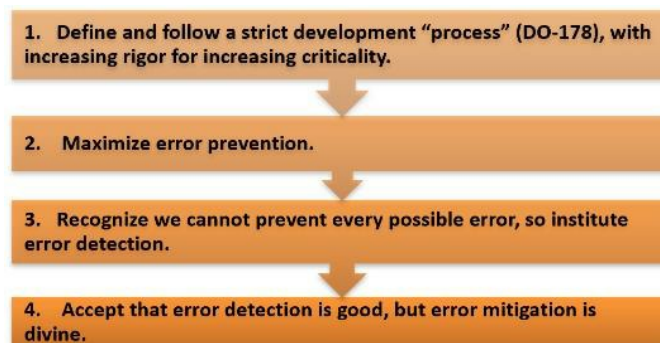
Providing guidance for acceptable use of new technologies could have been easily accomplished via AC’s or CAST papers instead of updating DO-178B. But the second objective, addressing savvy (or uninformed) developers who leveraged certain “shortcuts” available within DO-178B, was more problematic and required specific changes to DO-178B. Welcome to the next version, here today: DO-178C.

Safety-critical software has obvious needs for reliability. In fact, aviation reliability is defined in terms of “safety”, where formal safety analysis is performed before, and during, development of the actual software. In avionics, the safety strategy is based upon ARP-4761A (see the preceding chapter on SAE ARP4761A). However, in aviation, the software is NOT analyzed for safety: safety assessments are performed at the aircraft, system, and hardware levels; but not software. Why? Simple: “The probability of software failure is 1, as in 100%.” Now, other industries do their utmost to attempt performance of safety assessments upon software design and code. But not aviation. Simply put, software-based safety assessments are subjectively qualitative, not deterministically quantitative. But software design and implementation are directly related to safety. Instead of prescribing some form of safety assessment upon software itself, aviation instead imposes software engineering processes with verification and auditing of those processes to ensure provable adherence. But not software safety assessments. The best we can do at the software level is follow all of DO-178C’s 71 Objectives which should roughly equate to an equivalent level

of actual safety per system/hardware safety assessments. Those 71 Objectives within DO-178C are called “objectives”, not “subjectives”, because they can unambiguously and deterministically assessed.

While mythology surrounds the ability to expressly determine the actual reliability, or “safety”, of software, like most mythology the degree of fiction exceeds reality. But like mythology, we love a good story. So the safety story of software might say that we can analytically examine software to determine how safe or reliable it is. To some degree that is true: where algorithms or models exist we have varying degrees of ability to mathematically ascertain or predict the software’s performance. However, the ultimate goal, or measure of the ability to predict software’s reliability would be to answer the following question in the affirmative: “Can I prove this software is perfect?” Invariably, for all but the simpler software programs, the answer is, “No.” Why? Because software is only as good as its weakest link. Consider: what do you call software that is 99.9% perfect? Obviously, “0.1% imperfect.” For avionics then, what are the presumed odds that the 0.1% imperfection will manifest itself via an inflight anomaly? A near certainty.

Therefore, we cannot provably state that software is perfect. Thus aviation takes a different approach. Since we cannot readily prove our avionics software is perfect, we do the following instead:



Since DO-178B was released in the early 90’s, our knowledge of software development processes, techniques, and strategies for safety-critical software has vastly improved. In fact, if we had the same problem domain today as we did twenty years ago, we could almost guarantee that software errors could be reduced by over 99 percent. Don’t believe it? I’m old enough to remember automobile road-trips with my parents in the sixties and seventies.

Everywhere we'd see disabled cars: flat tires, overheated radiators, broken starters. Today? Most people don't know how to replace a flat tire, handle an over-heated radiator, or jump a car. This owes to a greater understanding of the automobile problem domain by manufacturers with significantly improved reliability of those mainstream components. It's typically the newer, less mature, technologies which experience problems within our automobiles.

Now compare that to avionics: the size and complexity of avionics software has increased exponentially. Very few avionics systems containing software have remained unchanged. Increasingly, software is used as both a product differentiator and a means to leap technical barriers. Software is made to do more on shorter development schedules. The solution? Adopting new software development and execution technologies. However, DO-178B could not have foreseen these new development and execution technologies. So while the commercial world embraced the new technologies, avionics was slow to follow.

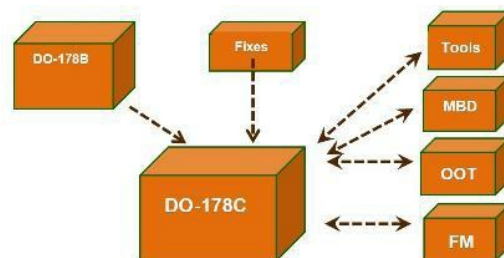
Worldwide certification agencies such as EASA and the FAA are staffed with sharp, hard-working individuals but typically they have less exposure to recent hands-on software development using these new technologies. Their approval of commercial avionics is a double-edged sword: if they do not approve, the avionics doesn't fly and both innovation and conquering avionics challenges are stifled. But approval means those systems are deemed safe by the very people who would be held partially accountable for any ensuing failures. The result? A true quandary and a difficult place to be for an avionics certification agency. But these same agencies could see the impending need for adopting new technologies and were not sleeping. They formed groups to study these new developments and devise strategies for adoption. They engaged industry experts and technical venues to both gather and disseminate information. They wrote issue papers. They coordinated across oceans. They educated their staff and industry. They adapted. And they knew there was no way DO-178B would survive the continuing technological evolution intact: an updated version, DO-178C, was needed.

Exactly what technological changes necessitated updating DO-178B? Compilers? Linkers? Microprocessors? Software test tools? Not exactly.

While those products continued to evolve and improve, they were not the principal catalyst for avionics development or certification change. Instead, the technologies that drove the need for DO-178C were those specifically able to better manage the #1 challenge facing avionics software development: the growing complexity of avionics software.

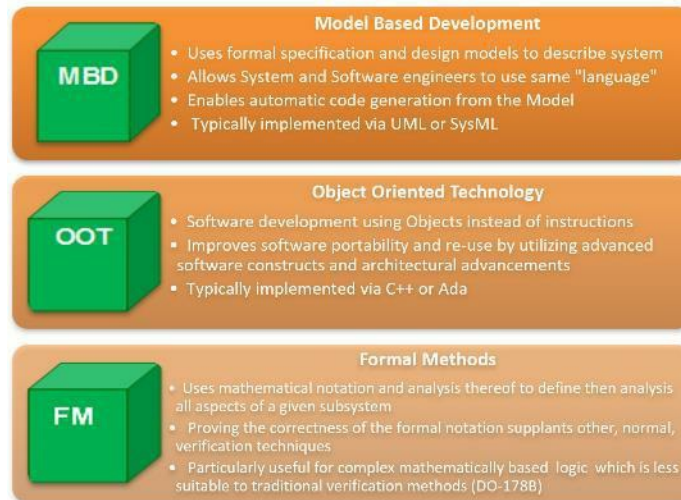
Fifty years ago airplanes had no “software.” Apollo 11, launched in 1969 as the first of the lunar landing series, had a guidance computer containing 36K of ROM and 2K of RAM. It’s a certainty that your battery wristwatch or personal microwave oven has greater capability. Twenty years later, commercial aircraft had over a million lines of source code. Today, aircraft have over 20M lines of code with vastly greater integration and complexity. Linear growth? Hardly: it’s exponential. At the same time, the convergence of military and commercial avionics coupled with greater need for software reusability meant that new software development techniques were needed. In particular, there were three technologies which avionics developers most wanted to leverage for improving their ability to manage this changing landscape: 1) Model Based Development (MBD); 2) Object Oriented Technology (OOT); and 3) Formal Methods (FM). These newer technologies, coupled with the need to correct several weaknesses within DO-178B and add new Tool Qualification aspects, yielded the new DO-178C as represented below:

Key Aspects Comprising DO-178C:



MBD, OOT, and FM as they relate to avionics software development are all-too-briefly summarized below.

Summary of MBD, OOT, and FM

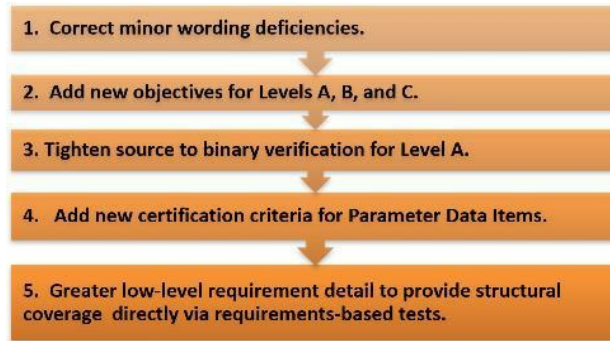


Clearly then, avionics development was embracing advanced tools, MBD, OOT, and FM technologies and new guidance needed to be formulated to enhance safe adoption. In fact, these new technologies were not directly added to the body of DO-178 but instead all-new supplements, one for each technology (Tools, MBD, OOT, and FM), were written. There were two reasons these new supplements were composed instead of simply adding the corresponding material to the body of DO-178:

1. By making them supplements outside the body of DO-178, other aviation related guidelines could reference and readily apply them, e.g. DO-XXX, including: DO-254 (Avionics Hardware), DO-278C (CNS/ATM Systems), DO-297 (Integrated Modular Avionics), etc.
2. DO-178C would have been five times larger, e.g. 600 pages total, if it contained the four supplements directly, thereby obfuscating readability.

In addition to the four technology supplements, changes and additions were made to the body of DO-178 itself, thus becoming "DO-178C". Unlike the revision of DO-178A to DO-178B which was quite a major philosophical shift twenty years prior, the revisions within the body of DO-178C are relatively minor. Those revisions are summarized below:

Key Revisions Within the Body of DO-178C



DO-178B admittedly had a number of minor wording inconsistencies, particularly when considered with latter guidelines including DO-248, DO-254, DO-278, and DO-297; these were largely corrected within DO-178C. One area that many DO-178B users hoped would be more fully clarified pertained to the term “robustness”; unfortunately, this term was left largely intact meaning there is subjectivity within robustness testing that each user must deal with. Indeed, there are so many different avionics domains and implementations that it is impossible to specify all possible manifestations of necessary “robustness.”

With a few notable exceptions, DO-178C did not significantly increase the rigor required for avionics software development. After all, the DO-178C committees were wisely well-represented by the user community and that group did not want to unduly increase costs, schedule, or risks. Kept intact was the clear delineation between “critical” software (Levels A, B, and C) and “less-critical” software, i.e.. Level D. If anything, that delineation between Level C and D was increased via the additional objectives added to Level C, and also the implication that Level C moved closer to Levels A/B because of the need for proof that software structures have more coverage via requirements-based coverage.

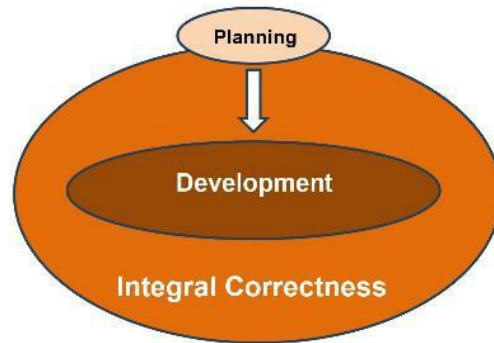
The quantitative change in objectives required for DO-178C levels is depicted below:

Level / Failure Condition	DO-178C # of Objectives	DO-178B # of Objectives	DO-178C Objectives Requiring Independence
A / Catastrophic	71	66	33
B / Hazardous	69	65	21
C / Major	62	57	8
D / Minor	26	28	5

E / No Effect	0	0	0 (Prove you're Level E!)
---------------	---	---	---------------------------

Note: For Level E, all that is required is to prove it's really Level E via a safety assessment.

DO-178C kept the same basic Planning/Development/Correctness process, but those processes must be updated to accommodate the changes described above.



DO-178C's three integral processes: Planning, Development, and Correctness:

That concludes the prelude to DO-178C. Additional details are found in subsequent writings. Several questions should be coursing through the reader's mind at this time, including:

1. *Did DO-178C make my job easier or harder?*
2. *What if I don't use MBD, OOT, or FM; must I still adhere to DO-178C and if I must, what's the difference to me?*
3. *I'm modifying a DO-178B application and it must be certified on a new platform to DO-178C; what will be the impact to me?*
4. *Unless DO-178C is the world's first perfect software guideline, what will be the most common areas of concern and challenges I should plan for?*

The above are highly relevant questions and the answer to each is "it depends..." The human need for explicitly black and white answers is elusive because the questions themselves are so colorful. But there are answers, and they are generally sound.

7

DO-178C Avionics Software

By Vance Hilderman

Reviewed by:

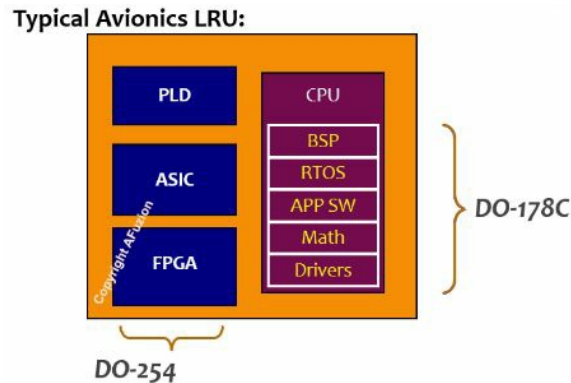
- Mr. Francois Guay, FAA DER

DO-178 has an innocuous title: “*Software Considerations in Airborne Systems and Equipment Certification.*” But it’s best not to judge that small 144-page book by its cover. In reality, “178,” as industry slang calls it, is largely considered the bible of avionics software development. Interestingly, since it was first developed in the 80’s (a time in which there was relatively little software in safety-critical systems), it has become the de facto embodiment for many of those systems. In fact, a careful examination of standards within military aviation, medical devices, railroads, automotive, and nuclear power will reveal striking similarities with 178. An accidental coincidence? Hardly. Was 178 the first such standard? That answer hardly matters, except that history is important. And why is history important in the case of DO-178? Because 178 has evolved via three subsequent iterations and it’s important to understand the reasons for those changes.

Remember the parable about the seven blind men and the elephant? Each touched a different part of the elephant and tried to describe what they were touching without knowing anything about the other parts of the elephant. An impossible task indeed, yet it’s an easy trap to fall into within aviation: aircraft and their systems are so complex that no single person can fully understand all the intricacies. In other words, without enlightenment, we’re all little better than the blind men. DO-178 attempts to lay a framework so that development personnel and certification authorities can work with full vision and leave residual blindness behind. 178 does succeed in providing such a framework. However, it doesn’t guarantee clarity of vision and certainly not perfection. What, avionics software is not perfect? Of course not. DO-178 isn’t perfect? Hardly. In fact, paraphrasing Winston Churchill’s

famous quotation concerning democracy, “DO-178 is the worst standard in the world ... except for all the others!”

Avionics systems are generally comprised of a Line Replaceable Unit (LRU) with allocation of logic to software (DO-178C) or hardware (DO-254) as reflected in the following figure:



In the book “Avionics Certification – A Complete Guide To DO-178 & DO-254” the principal author (the same as the author of this book you’re now reading) took 200+ pages to describe what DO-178 is. Describing what DO-178 is NOT can be much briefer: e.g., DO-178 is NOT:

- A prescriptive and rigid standard for software;
- An inflexible government standard with little bearing on market realities;
- A technical standard outdated as soon as it’s published
- A treatise on software development;
- A government standard written by people with little hands-on experience.

To be certain, DO-178 is the opposite of the list above. Truly, DO-178’s very success is generally recognized to be based on the facts (or at least perceptions) that it is:

- A flexible framework for the development of airborne software;
- Able to accommodate almost all types of systems regardless of application or complexity and which can evolve with technological advancements;

- A standard largely written by smart and successful technical personnel who, with their peers, will be bound to adhere to it and thus have a vested interest in its usefulness;
- A standard that avoids telling the industry a rigid method of “how” to develop and verify software and focuses primary on the “what” is required;
- A standard that modulates how many checks and balances (“what”) are minimally required depending on the criticality of the airborne software in that it reduces the cost commensurate with safety.

To begin understanding DO-178, it is necessary to understand “eco-systems”. For example, one cannot understand The Calculus without first understanding addition, subtraction, equations, and a few more mathematical concepts; The Calculus is merely a part of the mathematics eco-system. Similarly, DO-178 is a part of the avionics development and certification eco-system that includes a safety assessment process (ARP-4761), avionics system development (ARP-4754A), hardware design assurance (DO-254), environmental and EMI testing (DO-160), and many other relevant aspects. In many ways DO-178 is like that leg of the elephant; a necessary aspect of the elephant’s ability to stably exist: the leg doesn’t guarantee health but certainly lessens the probability of an early demise ...

DO-178 then starts with the premise (and for commercial avionics, a certification authority mandate) that the user applies DO-178 as merely one link within the avionics certification chain. Avionics will be neither safe nor compliant without a safe foundation to precede DO-178. What then comes before the application of DO-178? Typically a Project Specific Certification Plan (PSCP) will be developed, which defines the avionics eco-system for the avionics system including the applicability of DO-178. That PSCP-cited eco-system normally includes the performance of a formal Functional Hazard Assessment (FHA) per ARP-4761 followed by definition of system level avionics requirements per ARP-4754A. What happens if you initiate DO-178 activities without considering the preceding? Generally you will have enjoyed some nice practice activities, so you’ll “get” to go back and do it all again ... Truly, DO-178 requires a safe foundation with formally defined system requirements and any pretext to the contrary is foolhardy. In short, DO-

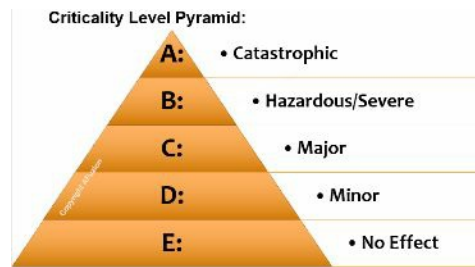
178 does not verify that the system requirements are good or bad; DO-178 assumes the system requirements are good and ensures they get transposed and verified to software adequately commensurate with the design level. However, avionics software quality can be no better than that of the apriori system requirements.

DO-178 is a relatively small document: vastly smaller than the documents you'll produce to prove you followed it. It's been said that the documents used to develop an aircraft would never fit inside that aircraft. Likewise, it's guaranteed that the documents developed to comply with DO-178 exceed the page count of the printed source code. Why? Because like any regulatory framework for safety critical software, there are plans, standards, specifications, designs, reviews, analysis, tests, and audits necessary to define, evaluate, and prove conformance. The same is true for military aviation, medical devices, nuclear power, railroads, and safety-critical automotive software.

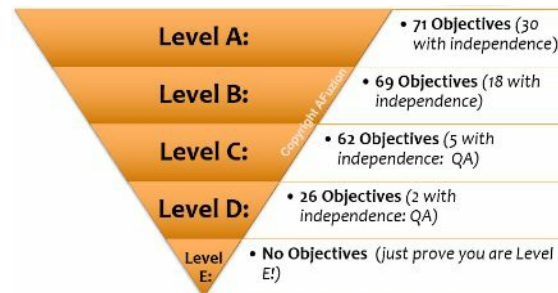
After a formal framework is in place to yield reviewed and controlled functional hazard assessments and system level requirements, it's time to consider the application of DO-178. Where to start? Well, you could borrow a copy of DO-178 and the ancillary documents and try to understand it. After you've read those documents several times, you've at least convinced yourself that they are interesting and important. But how do you really begin to apply it and where do you start when it seems like everything is strict, but nothing is really required? Now you need to really understand DO-178 ...

Development Assurance (Criticality) Levels.

First, you need to understand the Development Assurance Level (DAL) of the software you are developing for DO-178. The rigor applied to planning, development, and correctness of your software is directly associated with its DAL—often referred to as “criticality level.” There are five levels, with increasing rigor from Level E to the most stringent Level A, as depicted below:

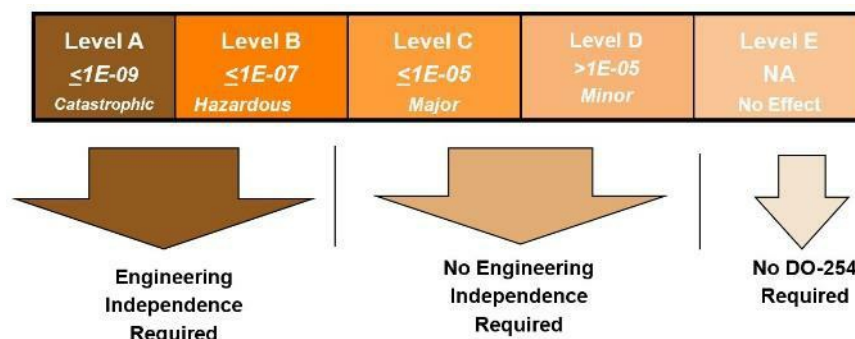


Remember, prior to developing software the DAL must be determined via the formal safety assessment process. The initial DAL is an output of the Aircraft Functional Hazard Assessment (FHA) and System FHA; the DAL could possibly be changed later as part of the continuous safety assessment process which includes feedback and reconsideration by Safety Engineering. As previously noted, the level of development and verification engineering rigor is based upon the DAL as depicted below:



Note that the assigned level and corresponding reliability is dependent upon type of aircraft; the following is for Part 25 aircraft, e.g. larger aircraft:

Engineering “Independence” per Development Assurance Level. (Note Quality Assurance must always be independent)



In the above figure, “Independence” is reflected as being required for DAL B, and increasingly more so for DAL A; this is verification independence, meaning a different person following a different process performs the

verification. Such independence does not need to be a different company or different location: in fact it's helpful if the independent verifier is co-located to maximize applied expertise and knowledge retention.

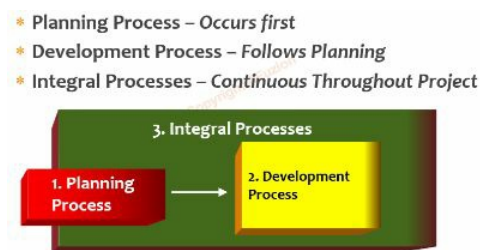
Surely you can simply choose your own assurance level for your software to make your work easier, correct? Of course not. The criticality level is essentially determined via the Safety Assessment process preceding the application of DO-178 plus application of applicable TSO's with consideration of actual operations. Why? Because the assurance level is related to the intended usage and design of the larger "system," of which software forms only a part. For example, secondary Nav is normally Level C but if that same VOR/ILS receiver is used in a Cat III zero feet decision height it may exceed the TSO minimum requirement. So, the level is not always cut and dry. The FAA took a stab at defining levels in Part 23 AC 23.1309-1D and its Appendix A goes where the other Parts do not go. But, each unique system has a DAL, and the software within that system has criticality levels equal to or less than the DAL of its system. Less than? Yes, it's possible for software, or a portion of the software, to have a criticality level less than its system. In fact, it is increasingly common for systems to have software with multiple criticality levels.

Why not make all the software Level A? Well, Level A software is likely higher quality than lower levels just as Level B generally has higher quality than Level C. So why not make everything Level A—the plane would be safer, right? Perhaps, but the cost would be higher. There are no free lunches in safety critical software. More safety costs more money. The avionics safety assessment process determines how much safety is required for each system in order to provide acceptable levels of safety; providing additional safety is simply not required. It's hard enough to comply with the minimum requirement...raising the bar above the minimum requirement, although noble, can put companies at financial risk. Nobility in this case is defined as to stay with the minimum requirement.

DO-178 has specific objectives based upon the criticality level of the software. Higher DAL's must satisfy more DO-178 objectives than lower levels. After the software criticality level has been determined, you examine DO-178 to determine exactly which objectives must be satisfied for the

software. Now you are ready for planning. This is where DO-178 is similar to building a house: you've performed geographic analysis to determine what type of foundation is required—that is your “safety assessment”. Then you need a Planning Process, followed by a Development Process. A concurrent Correctness Process is ongoing throughout both Planning and Development. Avionics software engineering under DO-178 is thus the same as building a house and follows the same three-phased process approach

DO-178, DO-254, and DO-278 have three integral processes: Planning, Development, and Integral Correctness:



As can be seen in the above figure: the Planning process comes first, and when complete is followed by a larger Development process. In the background the largest process, Correctness (or Integral Process of QA, CM, Verification, and Certification Liaison) is performed continuously. What is meant by Planning, Development, and Correctness? Here is a brief summary.

Planning Process. Before development, or before re-using pre-existing or legacy software, you need to plan your activities. Just like building a house, the building inspectors first need to inspect a set of plans followed by regular inspections of the house as it's being built: foundation, walls, electrical, plumbing, roof, and the finished building. DO-178 is similar. There are five plans and three standards associated with the Planning Process. The following figure summarizes key aspects of these five plans:

- Plans must precede Development activities
- Must provide proof that Plans are followed
- Plans address What, When and Who (and a little “how”)
- Plans typically approved by QA/CVE (and by DER in USA)

Quality Assurance (QA) is the principal signatory of the five plans. Why? Because QA is subsequently responsible for auditing engineer's compliance with those Plans; if QA has not reviewed and approved the plans, they could

not reasonably be expected to perform thorough audits against those plans. In North America, civil aviation under the Federal Aviation Administration (FAA) historically augments their certification activities by deploying Designated Engineering Representatives (DER's) who assist with certification and sometimes can even be formally give "approval" authority to essentially act as an FAA auditor/approver. In Europe, a less powerful role is sometimes used in place of DER's, that of the Compliance Verification Engineer (CVE) with less authority than a DER but still supporting QA. The five plans, often referred to as the "Keys of DO-178" are summarized below:

1.	2.	3.	4.	5.
PSAC	SQAP	SCMP	SWDP	SWVP

0 II	PSAC: Plan for Software Aspects of Certification
0 II	SQAP: Software Quality Assurance Plan
0 II	SCMP: Software Configuration Management Plan
0 II	SWDP: Software Development Plan
0 II	SWVP: Software Verification Plan

The Five Plans - Summary:

1. ***Plan for Software Aspects of Certification*** (PSAC): an overall synopsis for how your software engineering will comply with DO-178 including references to Safety and the System (as described in the preceding ARP4761A and ARP4754A chapters):

1.	2.	3.	4.	5.
PSAC	SQAP	SCMP	SWDP	SWVP

Plan for Software Aspect of Certification (PSAC)				
➤ 40-60 pages (strive for succinctness)				
➤ Overview of System, Architecture and Software				
➤ Discusses Safety, Criticality and Certification				
➤ References Other Plans and Artifacts				
➤ Help prepared by and Approved by DER				
➤ Submitted to and Approved by Cert Authority				

2. ***Software Quality Assurance Plan*** (SQAP): details how DO-178's quality assurance objectives including approvals, audits, and corresponding record-keeping will be met for this project:

1.	2.	3.	4.	5.
PSAC	SQAP	SCMP	SWDP	SWVP

Software Quality Assurance Plan (SQAP)

- Addresses the Role of QA throughout the development process
- Ensures the plans are coordinated and an integral part of the process... and followed!
- Ensures that Transition Criteria are adhered to
- Addresses the conformity reviews and inspections
- Provides guidance and timelines for audits/reviews by QA (Including checklists)

3. **Software Configuration Management Plan (SCMP):** details how DO-178's change management and baseline/storage objectives will be performed on this project:

1.	2.	3.	4.	5.
PSAC	SQAP	SCMP	SWDP	SWVP

Software Configuration Management Plan (SCMP)

- Established identification of data items
- Addresses the use of the CM system
- Defines criteria for Development CM
- Addresses the problem reporting and change tracking system
- Defines accessibility and authority for data
- Addresses security, maintainability and repeatability
- Addresses the reproduction of software (including binary) and all data items generated, and ability to repeat for manufacturing.

4. **Software Development Plan (SDP):** summarizes how software requirements development, design, code, and integration will be performed in conjunction with the usage of associated processes, standards, and tools to satisfy DO-178's development objectives:

1.	2.	3.	4.	5.
PSAC	SQAP	SCMP	SWDP	SWVP

Software Development Plan (SWDP)

- Defines the schedule, staffing, dependencies, deliverables, milestones and organizations
- Addresses the development environment and tools
- Addresses the lifecycle processes (Requirements, Design, Coding, Integration, informal developer testing (optional) and changes)
- Addresses RTOS, Partitioning, Previously Developed Software, Parameter Data Items (PDI's)
- Addresses CM and QA involvement (including Transition Criteria and deliverables).

5. **Software Verification Plan (SVP):** summarizes the review, test, and analysis activities, along with associated verification tools, to satisfy DO-178's verification objectives:



The Three Standards.

DO-178 (and DO-254 / DO-278) recognizes that requirements, design, and implementation (code) are fundamentally important and must be assessed. Assessment requires comparison of those requirements, design, and code to some pre-defined criteria. However, such criteria are inherently subjective. DO-178C has 71 “Objectives” which are objective because each can be unambiguously and completely assessed. However, requirements, design, and code for a particular system can have vastly differing assessment criteria. For this reason, DO-178 does not provide explicit objectives for requirements, design, or code. Instead, each project defines its own explicit criteria for such via three detailed Standards as follows:

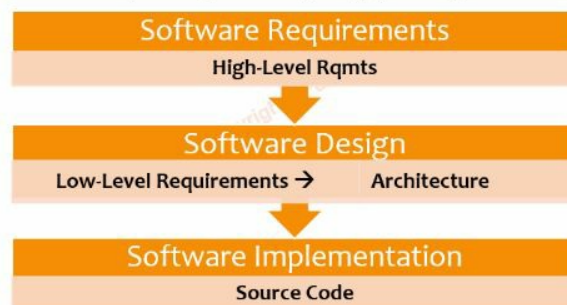
1. **Software Requirements Standard:** provides criteria for the decomposition and assessment of System Requirements into software high-level requirements and high-level to low-level requirements; including derived and safety related requirements. Note the low-level requirements (LLR's) can optionally be included in the Software Design Standard since Design includes completion of LLR's, and the software architecture.
2. **Software Design Standard:** provides criteria for completing the low-level requirements, defining and assessing the software architecture and design including both internal and external interfaces, data flow and control flow with coupling analysis.
3. **Software Coding Standard:** provides criteria for implementing and assessing the software source-code; typically these are expressed as a subset of MISRA-C.

The above five plans and three standards comprise the planning documents

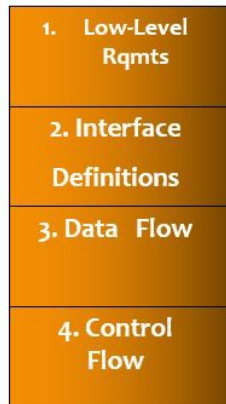
required for DO-178. Since Level D is the least rigorous of the five DO-178 levels, the three standards are not required; they are required for Level A through C. (Level E is practically not a DO-178 level since there are no required DO-178 objectives for that lowest level; you merely need a Functional Hazard Assessment (FHA) proving it's Level E hence not safety-related.)

The three standards (Requirements, Design, and Code) therefore codify project-specific criteria. Requirements are dealt with more thoroughly in a subsequent chapter within this book. Coding standards are generally just repurposed from other industry coding standards such as Institute of Electrical and Electronics Engineers (IEEE) or Motor Industry Software Reliability Association (MISRA) C-code standards. But the DO-178 software design is unique and covers project-specific software design techniques, processes, and documentation criteria to enable subsequent design verification. However, at what point do subtle software requirement details overlap with software architecture and is it possible to fully know all requirement details before initiating that software architecture? The answers lie in DO-178C's treating of the Requirement/Design/Code relationships as depicted below:

DO-178C's Relationship of Requirements, Design, and Implementation:

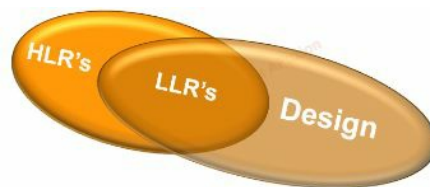


The key aspects of a DO-178C software design standard are summarized in the following figure:



Key DO-178C Software Design Aspects

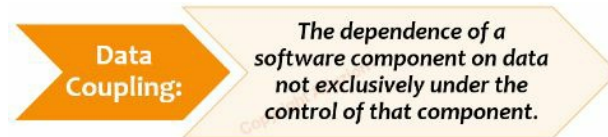
Since software design is the bridge between software requirements and code, safety-critical software design specification and verification comprise key traits of the DO-178C software development. A key facet of understanding DO-178C is the knowledge that the boundary between High-Level Requirements, Low-Level Requirements, and Design is “flexible”; each project simply defines its boundaries within the Software Requirements Standard and the Software Design Standard. Essentially, the Low-Level Requirements comprise the overlap of High-Level Requirements and Design as depicted below:



DO-178C's LLR's: Overlap of HLR's & Design

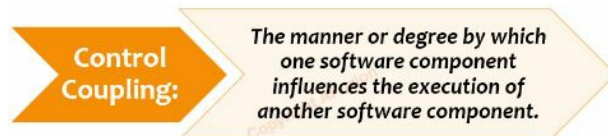
A Word On Data Flow & Control Flow.

DO-178C's design for DAL C through A must include documentation then analysis of the software data flow and control flow. This is performed to ensure determinism and assessment of data and control coupling. Data coupling is summarized in the following graphic:



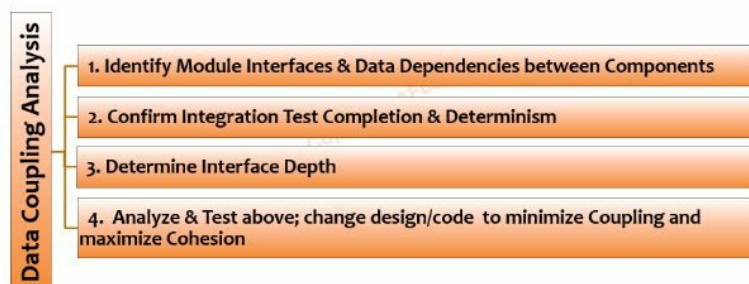
Data Coupling involves all the global variables and also those local variables which are passed down through parameter lists to lower-level components (example: function/procedure).

Control coupling is summarized in the next graphic:



Control Coupling occurs when one module passes data that is used to control the flow of execution of another module.

Since both data coupling and control coupling are assessable via DO-178C's software design documentation, coupling analysis is performed for the four primary reasons noted below:



Four Primary Roles of Coupling Analysis in DO-178

After your project-specific DO-178 Plans and Standards are reviewed and approved by your Quality Assurance department and the relevant certification authority, formal software development activities can proceed. Of course, if you have previously performed DO-178 or safety-critical software projects, you should already have basic software engineering plans and standards which you re-used to build the project-specific DO-178 plans/standards. In practice, it is common for preliminary software requirements and design to be initiated while the plans and standards are being developed to save time and clarify the details within the plans and standards. However, in theory, DO-178 postulates a classic waterfall model utilizing sequential planning, development, and verification processes. Why does DO-178 generally follow such a waterfall approach?

Because of History ...

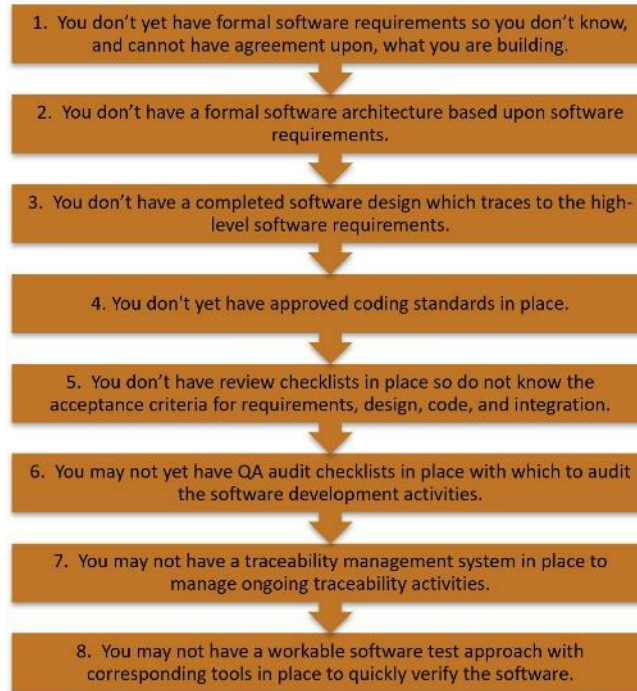
As noted in the preceding chapter, DO-178 was first composed in the 1980's, during the emergence of software “engineering” to accompany increasingly complex systems. Revised twice through the early 1990's, DO-178 generally reflected a number of software development best practices and philosophies prevalent at the time including CMM, military standards, and the now-infamous Waterfall methodology. Today almost no one uses a pure waterfall approach and in fact adherents today often use model-based development (MBD), particularly in conjunction with DO-178C and DO-331. Typical V-Model and even a safe subset of Agile is often applied with the latter more prevalent among lower DAL's.

While DO-178 allows virtually any deterministic, verifiable software development methodology, vestiges of the Waterfall are strongly imbued within the minds of many avionics development practitioners and auditors, particularly those with graying hair such as this author. What Waterfall attributes are commonly used?

- V-model accompaniment, with assessment of each phase's outputs
- Abundant documentation
- Sequencing and Transitioning of Requirements, Design, Code, Integration, Test

Now, back to avionics software development planning. You have now completed a strong Functional Hazard Assessment based upon ARP-4761. You have solid System-level requirements in place, based upon ARP-4754A, which consider the safety requirements. You have great plans and standards in place which are fully compliant to DO-178. Wow—you are ready to start developing software and writing code; quick, build a prototype so you can show management and the customer how proficient you are! Just one problem: you are not even close to being able to write real avionics software. Why? Let us count the reasons.

Reasons You Are Not Yet Ready to Write Software:



What, you can't just dream up some requirements for a nice house, buy some building supplies, and then invite all your friends over to start building your dream house while partaking in some pleasant beverages? Well, you could, but not if you want to legally occupy or resell your dream home someday. Avionics is the same way: while philosophical debates rage on whether or not DO-178 is formally "required", there is no debate that if you want to legitimately sell your avionics systems worldwide, the software needs to comply with DO-178. Where does that leave you? Simple, each of the reasons above should be addressed BEFORE you start writing software. Why? Well, there are regulatory and business reasons for such.

From a regulatory standpoint, it's important to be able to assess each artifact within the development process. Each development step has entry criteria and exit criteria, and those criteria are associated with discrete acceptance criteria. Where are such acceptance criteria defined? Within your development standards, (Remember the Requirements Standard, Design Standard, and Coding Standard?) and within associated checklists for each artifact. Checklists? Yes: aviation is based upon checklists. Every time an airplane prepares for takeoff the pilot and crew perform 'checks' based upon 'checklists'. When a commercial airplane (almost always with a pilot and a copilot) prepares for landing, the same thing occurs: the pilot may say

“Prepare for landing; flaps set to 10 degrees” whereupon the copilot affirms by reviewing the flap setting and announcing “Confirmed. Flaps set 10 degrees.” What just happened? The pilot was following a written landing checklist and the copilot was reviewing the pilot’s interpretation and actions based upon that checklist. Thus the copilot performed an “independent review” of the pilot’s required activities. Surely, the pilot and copilot have both memorized the landing checklists and do not need to actually follow a written checklist in the cockpit? Truly experienced pilots should have the checklist memorized. However, as we all know, memory can fail, particularly under stress, and checklists are written in advance and formally used during actual performance. Paper, digital, it doesn’t matter. What matters is that required activities have defined criteria which are specified in a checklist. Each checklist is reviewed, configured, and approved. That means there is one, and only one, valid version of each checklist for each operation. DO-178 is the same ...

A quick note on Objectives, Transition Criteria, and Checklists. DO-178 Objectives are identified in the annexes at the end of the official DO-178 document. Transition criteria act as acceptance criteria which correlate to those Objectives. Transition are associated with the Life Cycle model as opposed to the standards. Checklists are good but they need to be correlated to the Life Cycle Transition Criteria. Checklists are one level lower.

DO-178 uses checklists which you tailor to project specifics and standards; checklists also codify the success criteria applicable to DO-178 compliance. Checklists then satisfy the following attributes for each of your principal artifacts:

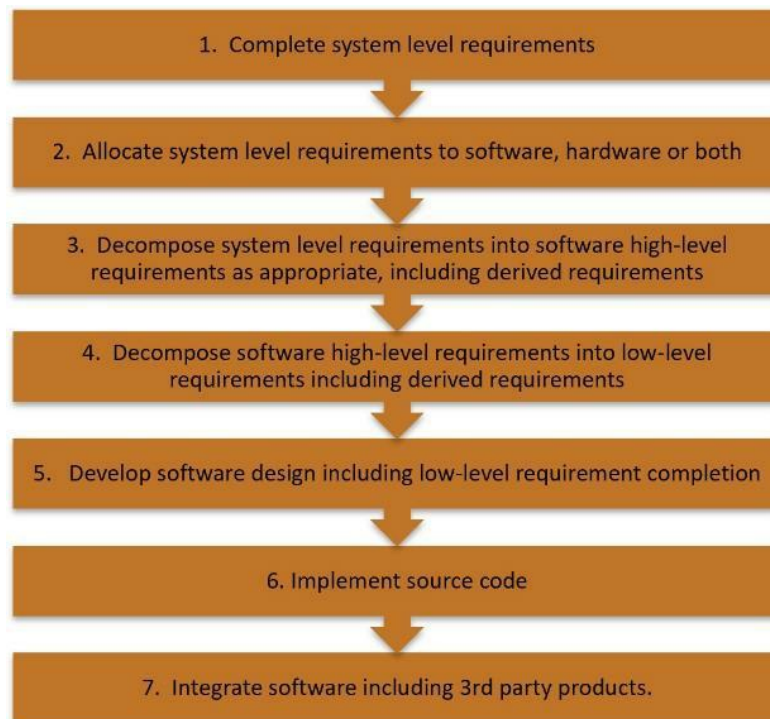
- Prove you have advance, formal details of acceptance criteria for each process and artifact
- Prove the review considered at least the minimum criteria, as contained within the checklist
- Show you can prove review and audit occurrence, with independence when required

What gets reviewed via checklists? Virtually everything, particularly items associated with required DO-178 objectives: Plans, Standards, Safety,

Requirements, Design, Code, Tests & Results, Tool Qualification(s), Suppliers, etc.

When your DO-178 Plans, Standard, and Checklists are complete, you are ready to begin software development. You should have passed the first of four mandatory certification meetings called Stage Of Involvement (SOI) since that SOI #1 assesses those very Plans, Standards, and Checklists. Following the classic Waterfall method, the following activities are performed, generally in order to show that you met the entry/exit criteria termed “transitions” within DO-178 parlance.

Classic DO-178 Software Development Activities



Of course each of the above software development activities will be performed according to the following previously reviewed and accepted documents:

1. Software Development Plan
2. Software Requirements, Design, and Code Standards
3. Corresponding Checklist for each activity, independently reviewed for applicable Level A and B software development activities.

In practice, today's complex safety-critical systems such as avionics rarely follow such a strict Waterfall sequence. Instead, model-based development, lean methods, prototyping, and evolving customer requirements are likely to be found. And the beauty of DO-178 is its non-prescriptive nature meaning flexibility. However, whatever method of software development you choose, including reusing legacy software, that method has to be detailed in advance via reviewed plans, standards, and checklists. Then you have to prove you actually followed such.

When you're done with software implementation, you have requirements that trace to code and that code can be traced back to requirements, and the requirements, design, and code are all documented and reviewed. You're done, right? Hardly. You are merely ready for the next Stage Of Involvement review, e.g. #2. What does SOI #2 do? It affirms your design complies with your plans/standards and such can be proven via review checklists.

SOI #2 should also affirm that your own project configuration management, and Software Quality Assurance (SQA) audits were conducted throughout your implementation. What is an SQA audit? DO-178 is about advance planning then proof (checklists) that the work was accomplished to plans and standards. However, the higher the criticality level, the less sole trust is allowed for the author because lives are at stake. The FAA has a written Order (FAA Order 8110-49, Chapter 3) that dictates the Level of FAA Involvement (LOFI) based on Criticality and takes into consideration many other factors such as the developer's software certification experience. The higher the criticality, the higher the risk the certification authorities perceive and since they have limited resources, they tend to focus on those higher risk projects particularly when being performed by newcomers. In fact, DO-178 has multiple levels of individuals who are involved with reviews or audits:

1. A technical review is performed upon a completed artifact ("ready for review") to assess its compliance to plans, standards, and checklist. Higher criticality levels require review independence, e.g. by a technical individual uninvolved with authorship.
2. SQA audits are performed to assess adherence to process, not necessarily technical compliance. Were the processes used during authorship compliant with plans and standards, and were transition

criteria (appropriate use of entry and exit criteria for each process) adhered to?

3. Certification liaison (such as Designated Engineering Representatives, DERs, in the United States) audits assess prior technical reviews and SQA audits.
4. Certification Authority (such as EASA or FAA) audits the above.

As can be seen, SQA within avionics is different than other industries. Where those industries often have SQA performing technical reviews and tests, DO-178 focuses SQA upon two primary activities; first, define or approve the project plans and standards; then assess via audits whether or not those plans and standards were properly followed. DO-178 does not dictate how SQA does their work; it simply wants SQA to hit the minimum set of objectives and maintain that position throughout the process. Simple.

After SOI #2 has been approved, the software implementation is largely done though there are always minor requirements changes and bug fixes. But full software testing needs to be performed. While software testing in other domains is often considered “art”, the problem with “art” is that it is subjective: we cannot agree on a definition for art but we all “know it when we see it”. The problem is that humans never fully agree on what is good art versus bad art. Same as music, food, weather, etc.: those likewise are subjective. So DO-178 transforms the art of software testing into a science, by requiring detailed tests of both the software requirements and of the code. DO-178 software testers must be proficient at writing tests to assess whether the logic meets the full extent of the written requirements. DO-178 also introduces a concept of “are we done yet”. If left to perfection, software testing could take on virtually infinite combinations and permutations. DO-178 limits that even for the highest criticality. Also, the higher the criticality level of the software the more that software code must be shown to be covered by tests, e.g. structural coverage assessment.

Classic DO-178 Software Verification Activities

Have you ever heard the statement “I am a V&V Engineer; some consider me an expert.”? Software testing is always part art and part science. In DO-178, software testers truly need to be expert in software design and code. In non-

avionics domains, it's common for "software" testers to know very little about the intricacies of software design and code. Not so in DO-178. First, a little about "V & V". As everyone knows, V & V is a common moniker for Verification and Validation. Unfortunately, the real meaning of V & V, particularly within avionics software, is often misunderstood. Hundreds of books have been written on software V&V. So the following paragraph does not give justice to this very involved topic.

Verification is the assessment of a process result to determine if the output meets specification. For avionics software, what then is "verification"? Technical people work best with quantitative equations (if you're not "technical", then why are you reading this?!?). So, avionics verification has the following equation:

$$\mathbf{V} = \mathbf{R} + \mathbf{T} + \mathbf{A}$$

Where:

V is "*Verification*"

R is "*Review*"

T is "*Test*"

A is "*Analysis*"

(Note the "A" for "Analysis" is intentionally shown small as analysis is only used when the combination of Review and Test does not fully verify the requirement.)

What does this equation mean for avionics software? Here's the answer:

- Human-created artifacts should be Reviewed (via Checklists)
- Software Requirements and Code should be Tested
- If the Test is not conclusive by itself, additional Analysis is required.

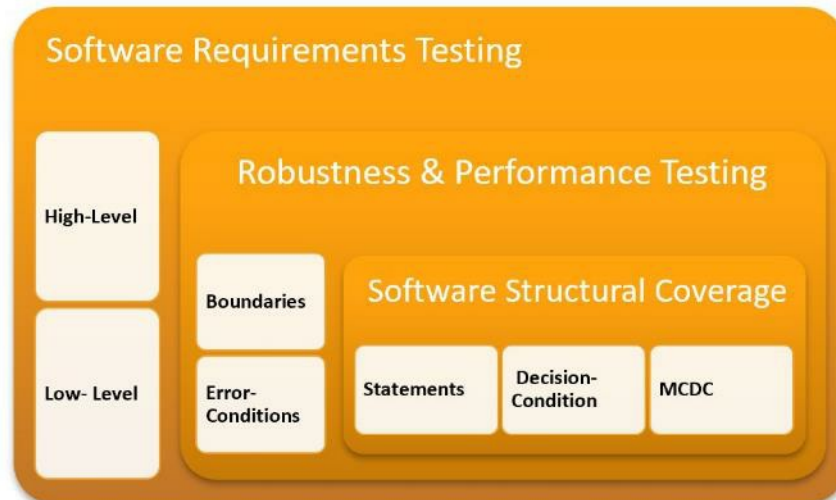
Of course, the degree of reviews, tests, and analysis is wholly dependent upon the criticality level of the software as will be shown below. However, that summarizes Verification, but what about Validation? Remember: verification assesses an artifact to determine if it meets specification. For software verification, the specification is the "requirements". However, what if the requirements aren't great? That's where Validation comes in:

Validation assesses the Requirements to determine if they are great, where great means clear, concise, correct, complete, and verifiable. Garbage in, garbage out. The most important input to avionics software quality is the Requirements. So, must avionics software requirements be validated? Actually DO-178 does not mandate such because validating software requirements is considered subjective without a completed system and DO-178 focuses upon objective criteria. Officially, requirements must be validated at the Hardware and System levels, per DO-254 and ARP-4754A respectively. Unofficially, best practices should include validating software requirements during the requirements review process, e.g. they are correct, and they are complete.

As mentioned above, software testing is part art, and part science. And like software development, perfection is impossible. However, DO-178 was written by industry professionals with only minor government oversight, so the focus is upon reasonable cost-effectiveness. It's easy to spend more resources testing the software than were used to develop it. Over the avionics product lifetime, it's a certainty more time will be spent testing than developing.

What then comprises reasonable, cost-effective software testing? First, ensuring you have good software requirements, and those requirements are fully verified. Next, additional robustness testing which tries to find errors in the logic via stress/performance testing, exercising boundary and invalid values, and executing all transitions. Finally, structural coverage assessment to ensure that all logic which could potentially execute on the plane is verified and behaves as intended. Top-to-bottom traceability is formalized to prove all documented requirements are verified, and bottom-to-top traceability is included to prove there is no undocumented logic. Be careful readers: bottom up traceability is not simply inverting the matrices. It means that each segment of the code needs to trace up to a requirement (high, low, or derived). The segment of the code correlates to the depth required by the criticality.

What is the scope and relative effort of DO-178 Testing? A picture may help:



DO-178 Software Testing Relative Effort - Optimal (Level A)

First Focus: Tests Based Upon Requirements (E.g. Functional Testing)

First, testing is based upon requirements for DO-178 (the latest version, DO-178C, goes further by stating all major code segments should trace to at least one requirement). Since DO-178 does not provide subjective thresholds for requirement granularity, testing of requirements is dependent upon the requirements themselves. However, DO-178 compensates for potentially weak requirements by requiring, for Level A through C, software to undergo additional robustness testing and structural coverage assessment. If you have good requirements, testing those requirements should typically yield 90% coverage of the requisite robustness cases and 80% of the code for Level B. Why? Because good requirements provide good detail for low-level functionality and potential robustness conditions; such can be gleaned from the requirements thus test cases can cover 80-90% of the necessary conditions just by assessing the requirements and writing test cases for them. However, if you have weak requirements, then writing test cases from those weak requirements may only yield 50-60% of the requisite coverage; in that case, you will discover the missing requirement detail during testing structural coverage activity (though not required Level D and E) and be required to go back and add requirements. Clearly, like most things in real-life, it's much more cost-effective to do it well the first time instead of going back to improve it and do it again. So, if there is a lesson learned to be shared here: invest in good requirements up front and support them with structured

reviews and checklists.

It's imperative to note that DO-178's five criticality levels call for significantly more software testing as the criticality level increases. For the software development, criticality level has less impact but not so for testing as the following figure summarizes the testing required per DAL.

It's imperative to note that DO-178's five criticality levels call for significantly more software testing as the criticality level increases. For the software development, criticality level has less impact but not so for testing as the following figure summarizes the testing required per DAL.

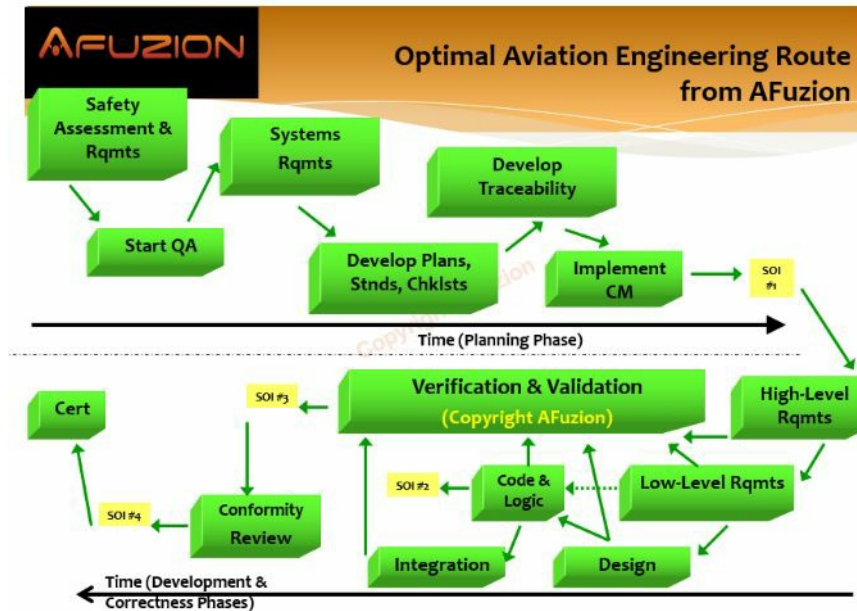
Major Testing Differences Between DO-178 Criticality Levels:



After testing, SOI #3 assesses the completeness of the verification activities. Then SOI #4 accompanies the Conformity Review which addresses compliance to all applicable DO-178 objectives, including interim “changes”.

Putting it All Together.

The following figure comprising an optimal DO-178 implementation which this author uses in most of his teaching classes is called the Green Snake:



An Optimal Implementation Route for DO-178C

As depicted above, the Green Snake begins in the upper left with the Safety process then proceeds left-to-right as the system then software planning processes are performed culminating in a review of the plans and standards (SOI-1) prior to the official start of development. Then, on the bottom half right-to-left, development then testing then conformity are performed. Of course “Certification” occurs only in the context of the system and aircraft with SOI-4 generally required to be completed prior to flight testing so that all safety-related aspects can be affirmed prior to such flight testing. Why is this snake displayed in green? Simple: it’s this author’s opinion that while DO-178 is flexible and non-prescriptive, the above Green Snake path optimally reduces overall cost. This author is American and American money is green: to save money, follow the Green Snake.

8

DO-254 Avionics Hardware

By Vance Hilderman

Reviewed by:

- Mr. Francois Guay, FAA DER

DO-254 has been called “DO-178’s Little Sibling.” Like many little brothers and sisters worldwide however, the term “little” is often wrong and in a few areas the relationship itself is suspect. DO-254, or more properly, “*Design Assurance Guidance for Airborne Electronic Hardware*,” was created as an obvious response to three simultaneous and related events:

1. Firmware was playing a larger role in avionics, with developers rapidly leveraging increased silicon-based designs;
2. The avionics firmware development process was unregulated, with certification performed after-the-fact at the system, black-box level; and
3. Evidence was surfacing which indicated some manufacturers were intentionally moving logic from “software” to silicon-based “firmware” to avoid software’s DO-178B mandate.

While embedded avionics software engineering made huge strides and inroads in the eighties and nineties, silicon-based firmware development was considered an informal adjunct art form. But what is “firmware”? When does “soft” become “firm” become “hard”?

DO-254 – A Brief History.

Two decades ago, silicon-based “firmware” was relegated to specialized functions within avionics as compared to its highly varied and rapidly expanding role today. There were multiple reasons early firmware was more limited in aviation:

1. Firmware development tools provided limited flexibility compared to

software;

2. Firmware was difficult to change once burned or loaded into silicon devices. Easier to use FPGA's didn't come into the forefront of programmable devices until the mid-1990's and more conservative aviation was slower to adopt;

Firmware was considered a less-desirable option than software due to a seemingly more difficult debugging and update process, largely due to the preceding #1 and #2.

While there was truth in each of the aforementioned reasons, smart engineers coupled with more capable development environments assured the evolution and adoption of firmware within avionics. In particular, great strides in Field-Programmable Gate Arrays (FPGA's) brought such firmware to the forefront of aviation. With FPGA's, all of the aforementioned restrictions on firmware adoption were dramatically reduced. FPGA's increasingly leveraged very modern development tools, were easy to update, and allowed for potential flexibility and execution speed advantages over software-based logic.

Certification (both civil and military) authorities require assurance that avionics hardware is developed with consistent and deterministic processes replicating those of the software development lifecycle with key differences noted herein. However, traditionally hardware could be exhaustively tested since it was considered "simple". For such simple hardware, all of the input/output combinations could reasonably be tested and therefore hardware was tested as such as part of systems testing. However, silicon-based firmware can be just as complex (often more so) than software and all input/output combinations cannot be reasonably tested and therefore it is considered "complex", not "simple". More on complex and simple hardware later. But, DO-254 requires hardware development processes consisting of Hardware Requirements, then Conceptual Design, then Detailed Design, then Implementation, then Validation and Verification, then Production Transition; all throughout these processes are background processes of configuration management and process assurance (similar to software's quality assurance). Design Assurance Level (DAL) C and D hardware development is performed black-box, meaning primarily Requirements, Tests of Requirements, Traceability, and Reviews of the requirements and tests are

required, inclusive of Process Assurance. However DAL A and B hardware add white-box, meaning standards, reviews, and tests for actual logic design/implementation.

Traditionally, hardware development differed from software development in the following ways:

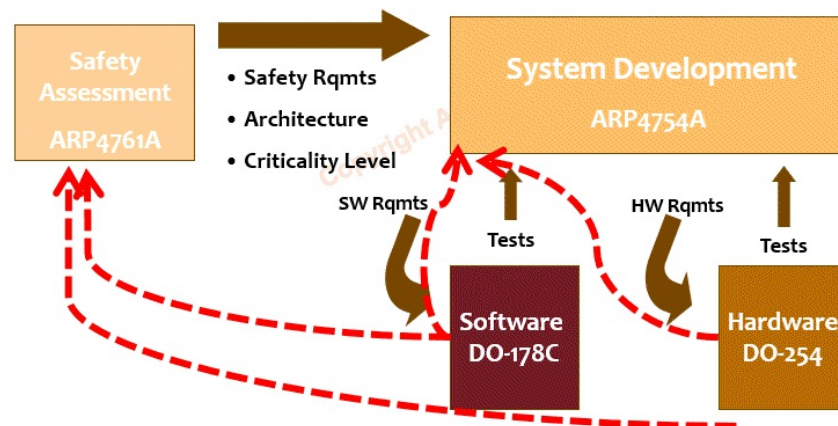
- Software complexity outpaced that of hardware.
- Hardware development teams were more senior, and smaller, than software.
- Much of the hardware could be gradually prototyped with testing done in parallel.
- Many aspects of hardware development could be inspected or exhaustively tested.

But with the advent of modern Application Specific Integrated Circuit (ASIC) technology, then FPGA's, those differences between software and hardware dissipated. Therefore the focus of DO-254 has evolved to more of a focus upon complex electronic hardware ("CEH", e.g. silicon-based logic), with Certification Authorities Software Team (CAST) Memorandum #27 followed by A(M)C 20-152A further refining DO-254's interpretation and focus.

As a result of this evolution (or almost "revolution") in silicon-based logic, avionics developers increasingly exercised the choice of implementing logic via silicon instead of software. However, DO-178 did not strictly apply to silicon based logic and there was no regulatory counterpart for such; thus silicon-based logic could potentially avoid the certification rigor associated with software, even though the logic may have a nearly identical purpose between software and silicon. Thus the need for DO-254: to ensure that hardware and logic developed specifically for avionics purposes underwent a similar development and certification regimen as if it had been implemented via software using DO-178 ...

DO-254 (and its counterpart ED-80 in Europe) was originally intended to cover all electronic hardware including Line Replaceable Units (LRU's), Circuit Card Assemblies (CCA's), Commercial-Off-The-Shelf hardware

components, Simple Electronic Hardware (SEH), and Complex Electronic Hardware (CEH). It is important to remember that DO-254 was written as a counterpart to DO-178B, not DO-178C, and in a period when “hardware” meant everything “physical” within a system as opposed to today’s emphasis upon hardware’s more prevalent complex silicon-based logic. Later publications including FAA Advisory Circular (AC) 20-152, Certification Authorities Software Team (CAST) memorandum 27, and EASA CM SWCEH-001 clarified and in some cases reduced the scope of DO-254 and ED-80. Particularly in the USA, DO-254 was employed with a greater focus on custom CEH versus the other hardware aspects cited above. Why? Because CEH employs “logic” (HDL, VHDL, etc.) and logic resembles software code: provable safety and quality require planning, standards, and incremental assessments because after-the-fact verification is not as effective as “process” to ensure that quality. Also, avionics developers in the USA had an earlier application of ARP4754A to avionics which required formal processes, requirements, and testing of hardware irrespective of DO-254. European, South American, and Asian counterparts were originally less likely to apply ARP4754A, so they used DO-254 to help ensure certification coverage of hardware. Today, DO-254 is an integral part of the Avionics Development Ecosystem as depicted in the following figure:



DO-254's Role Within the Avionics Development Ecosystem

Modern electronics employ sophisticated hardware and software whose quality cannot be fully assessed solely by end-item testing. Thus the processes used to engineer the associated hardware and software are

important and those processes must be planned, assessed, and controlled; DO-254 describes those processes for avionics hardware. As depicted in the preceding figure, avionics hardware development adheres to DO-254 but also considers the safety assessment and system development activities which surround that avionics hardware process. There is continuous involvement of both Safety engineering and Systems engineering so that adherence to, and feedback of, safety requirements and systems requirements is managed inclusive of bi-directional traceability.

DO-254 is:

- A flexible framework for the development of airborne hardware containing avionics-specific functionality.
- Able to accommodate almost all types of hardware ranging from sensors, multiplexers, switches, and aggregated simple silicon devices, in addition to full-featured FPGA's and ASIC's.
- A guideline which tries to cover a nearly infinite spectrum of hardware varieties thus lacks specificity for particular projects.

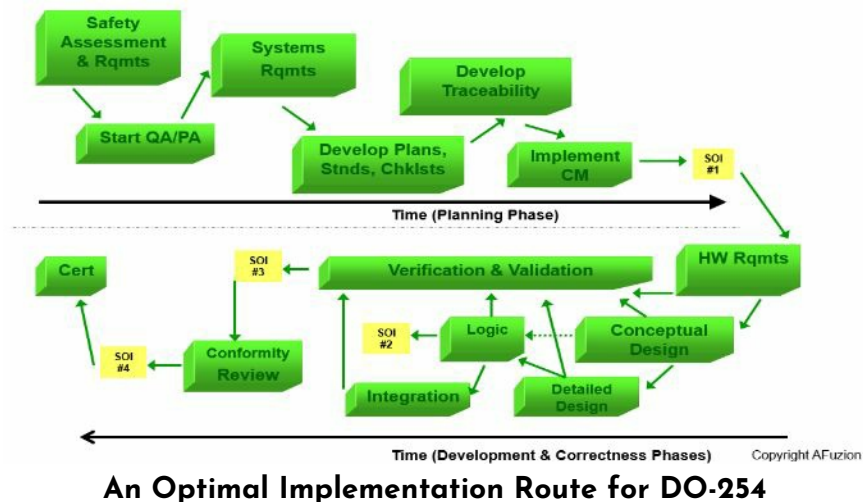
Like software, the term “airborne electronic hardware” from DO-254's title is wide-ranging. At the beginning and end of the day, hardware is part of a system or more specifically an aviation ecosystem. Therefore, for civil aviation, DO-254 is preceded by a safety assessment per ARP4761/A and an avionics system development process per ARP4754A; military aviation is gradually adopting a similar (in some cases identical) safety/systems process. And the hardware itself will typically be required to undergo environmental testing via DO-160. Therefore, DO-254 is merely one link within the avionics certification ecosystem. Avionics hardware cannot be provably safe nor compliant without this ARP4761/A and ARP4754A safety/systems foundation which both precedes and accompanies DO-254.

The DO-254 process employs detailed project-specific planning followed by continual assessments and process feedback loops to ensure defined hardware development processes specified in those Plans and Standards are followed. The preceding two chapters on DO-178C explained the need for and content of the five plans plus three standards. The figure and text below depict DO-254's similarity with DO-178C:

DO-254's Three Processes & Similarity with DO-178C



For an overly simplistic view of the DO-254 lifecycle process (without depicting requisite feedback loops, changes, reviews, etc.), this author's opinion is that the following comprises an optimal DO-254 engineering development lifecycle; note the similarity with the "Green Snake" for DO-178C which was previously depicted:



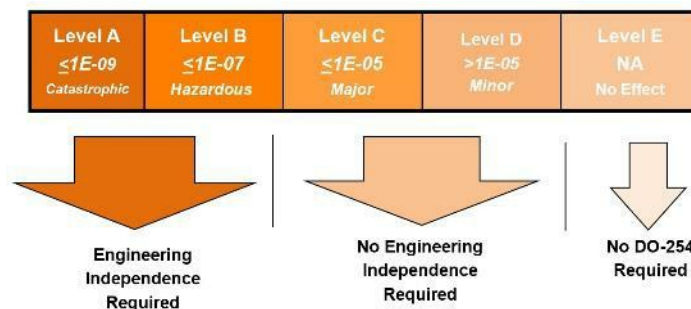
Starting with Safety then Development Assurance Levels.

Modern civilizations (and a few ancient ones) propounded a theory of "equality" amongst citizenry. While applicable for real life, the avionics ecosystem is anything but. Consider a car: while modest debate could ensue regarding the relative contribution toward safety of different systems such as steering and brakes, no one would seriously propose the car radio is equally important. In avionics, the differing contribution towards flight safety between systems is more pronounced and formally regulated. A safety assessment per SAE ARP4761A is commonly performed for each system where the safety assessment and functional hardware analysis postulate "what

are the potential failures and what is their impact?”. This safety assessment directly affects system and hardware design including redundancy, fault detection, mitigation, etc.; in fact, the safety assessment is used in the determination of system and hardware architecture and Development Assurance Level (DAL). Each unique system has such a DAL and the hardware (termed Item Dal) within that system most often has a criticality level equal to that of the system (termed Function DAL).

DO-254 has specific objectives based upon that hardware’s DAL. There are five DALs associated with airborne avionics systems, noted as level A through E, with level A being the most stringent. For software, under DO-178C, the differences between levels are greater than they are under DO-254 and each software DAL has distinctly discrete Objectives ranging from 26 objectives for DAL D to 71 objectives for DAL A. Overly simplified, DO-254 levels A and B are nearly identical with strict criteria applied to the engineering processes associated with each line of hardware logic; levels C and D are less rigorous and focus upon hardware black-box requirements/testing and lack consideration of hardware logic development and test. Level E requires no additional hardware design certification under DO-254 . The rigor applied to planning, development, and correctness of the hardware is directly associated with its DAL, often referred to as “criticality level.” These five levels with increasing rigor from Level E to the most stringent Level A, are depicted below; note the similarity with DO-178C

**Engineering “Independence” per Development Assurance Level.
(Note Quality/Process Assurance must always be independent)**



“Independence” refers to the Integral Processes, e.g. Correctness: Verification, Validation, and Process Assurance. These Integral Processes must be shown to have used evaluation by an individual other than the

originating engineer where that individual also performs their evaluation using a different process. For example, verification includes reviews, tests, and analysis.

As previously noted, the DAL greatly affects process rigor applied to hardware via DO-254. The following graphic summarize key differences between the DAL's for DO-254:

DO-254 Aspect	Level A	Level B	Level C	Level D
Independence Level	Yes	Yes	No	No
Necessity Hardware Requirements	Yes	Yes	Yes	Yes
Necessity of Conceptual Design	Yes	Yes	No	No
Necessity of Detailed Design	Yes	Yes	Maybe	Maybe
Pin Level Coverage	Yes	Yes	Yes	No
Statement Structural Coverage	Yes	Yes	No	No
Decision/Condition Structural Coverage	Yes	Yes	No	No
MCDC Structural Coverage	Optional – FAA Advised	No	No	No
Advance Analysis Method	Yes	Yes	No	No
Configuration Management	Tight	Tight	Medium	Low
Hardware Archiving & Hardware Reviews	Yes	Yes	Low	No
Requirements Correlate to Target Device	Yes	Yes	Yes	Yes
Architecture & Algorithms Verification	Yes	Yes	Yes	No
Logic/Model Reviews	Yes	Yes	Yes	No
PA Transition Criteria	Yes	Yes	Yes	No

Key DAL Aspect Differences for DO-254

Again, the DAL of the hardware is usually, but not always, the same as the System DAL. Why? The system contains hardware which is necessary to perform its functionality and that hardware is expected to directly correlate to the system's contribution toward flight safety. However, this principle cannot always hold. For example, the author of this paper worked on one DO-254 project where the System was DAL B, the Software was DAL B and DAL D, but the FPGA was DAL C; in that case the FPGA was used to merely translate data packets which were checked both upstream and downstream via software. Such is the interesting thing about rules: for each one, an exception can be found.

Process Assurance is like Quality Assurance but with a larger scope including auditing of hardware suppliers and manufacturing transition processes. Process Assurance has five primary activities as depicted in the following diagram:

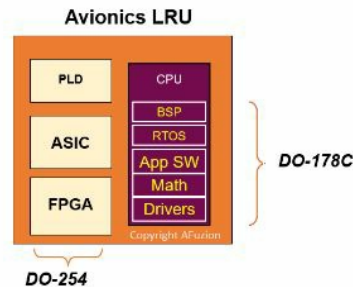


DO-254's Process Assurance: Five Key Roles

Process Assurance differs from software's quality assurance because hardware's Process Assurance must audit hardware suppliers and ensure subsequent system manufacturing processes are documented, repeatable, and conform to plans. Since hardware's Process Assurance has these two additional roles versus software's Quality Assurance, it has the different name "Process Assurance".

Scope of DO-254

DO-254 applies to most all avionics hardware, however more recent interpretations and application of DO-254 focus upon Complex electronic hardware as noted previously. Why? Because the Simple hardware is definable via black-box requirements and those requirements (and thus all the Simple functionality) can be tested at a black-box system level for example as already required under ARP4754A. For Simple hardware, the significant additional cost of detailed planning, detailed design, and low-level verification activities prescribed by DO-254 provide little added-value hence are normally not required unless a specific certification authority deems them necessary in a particular instance, for example because ARP4754A was not otherwise applied. Therefore, the most common application of DO-254 is depicted in the figure below which also elucidates the scope of DO-178C:



DO-254's Planning Process

Wise persons know “What gets planned, gets done”. Wiser avionics authorities know “What gets planned thoroughly can be assessed more thoroughly”. Accordingly, DO-254 requires a detailed planning process consisting of five Plans and four Standards:

1.	2.	3.	4.	5.
PHAC	HPAP	HCMP	HDP	HVVP

1. **PHAC:** Plan for Hardware Aspects of Certification
2. **HPAP:** Hardware Process Assurance Plan
3. **HCMP:** Hardware Configuration Management Plan
4. **HDP:** Hardware Development Plan
5. **HV&VP:** Hardware Verification & Validation Plan

Plus 4 Standards: Hardware Requirements, Design, Archival, V&V

PHAC .

The Plan for Hardware Aspects of Certification (PHAC) is the foundational planning document of an avionics hardware system. The PHAC provides an overview of the avionics system’s hardware, safety criteria with respect to DAL, and planned certification activities as noted in the following figure:



HPAP .

The Hardware Process Assurance Plan (HPAP) describes the planned process assurance activities with respect to planning, audits, transition criteria, and

conformity reviews. Note that DO-178C had “Quality Assurance” (QA) whereas DO-254 terms that activity “Process Assurance” (PA). The primary PA activities are depicted in the following diagram:



Primary DO-254 Process Assurance Activities

Note that two of the activities in the above figure are highlighted in yellow: those are unique to hardware's PA role and not formally required in DO-178C's QA role. The HPAP primary topics are summarized in the following figure:



HCMP .

The Hardware Configuration Management Plan (HCMP) describes the planned configuration management activities to ensure hardware baselining and change management activities provide for configuration replication/ repeatability as noted in the following figure:



HDP .

The Hardware Development Plan (HDP) describes the planned hardware requirements, conceptual design, detailed design, implementation, and integration activities as noted in the following figure:



HVVP .

The Hardware Verification & Validation Plan (HVVP) describes the planned hardware requirements validation and hardware verification (reviews, tests, and analysis as applied to requirements, design, implementation, and integration) applicable to assess the hardware. Remember: “Validation” ensures requirements are complete and correct whereas “Verification” ensures the implementation meets those requirements. Officially Verification and Validation are equally important, however this author always advises that Validation is more important: if the requirements are not correct and complete, then it doesn’t matter if they’re verified! Remember, the HVVP describes the means to verify both the human hardware engineering decisions which went into the hardware, and the resultant end-hardware. Hardware development, particularly for silicon, requires a plethora of tools to design, simulate, synthesize, and manufacture the silicon and circuit boards. All the human decisions made in using those tools must be documented and verified, in no small part so that subsequent engineers can evaluate those decisions and

ultimately change the hardware if defects or new capabilities are needed. The Hardware V&V Plan is where those V&V activities are documented and it is summarized in the following figure:



The Standards.

DO-254 is wisely silent on prescribing particular avionics hardware development criteria since no one-size-fits-all recipe could be practical to the myriad forms of hardware. Instead, DO-254 requires the hardware developer to define detailed Standards for the four key engineering phases and then assess subsequent compliance to those Standards. That assessment is performed by verifying the associated hardware artifacts using retained and configured checklists which capture the criteria contained within those Standards. The four key DO-254 hardware engineering phases requiring Standards are:

- **Hardware Requirements Standards**, providing criteria for identifying, specifying, and tracing the applicable functional and derived requirements.
- **Hardware Design Standards**, providing criteria for documenting the conceptual hardware design then the detailed hardware design including all internal and external interfaces.
- **Hardware Verification & Validation Standards**, providing the techniques, scope and criteria for validating, reviewing, testing and analyzing the hardware requirements and implementation.
- **Hardware Archive Standards**, providing the techniques for capturing and storing hardware artifacts with sufficient granularity to enable replication of all such artifacts in the future.

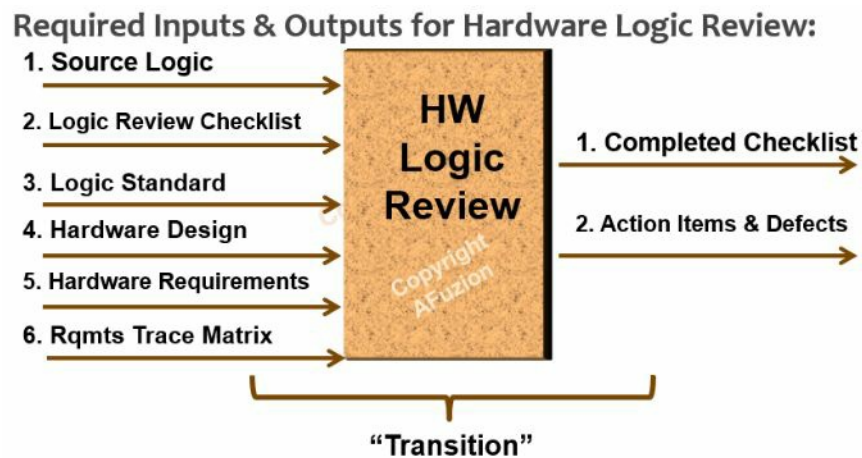
A good way to understand the application of DO-254's four standards is to consider that DO-254 itself covers the more objective (measurable) aspects of the hardware development lifecycle, whereas certain aspects are more subjective yet equally important. Those subjective areas are hardware requirements, hardware design, hardware archival, and hardware V&V. Since each avionics project must have its own processes and assessment criteria for these subjective areas, such are specified within the four project-specific Standards thus making these “subjective” aspects “objective” for each project.

While most of the aviation guidelines are self-contained, DO-254 is unique: because it was originally intended to address all non-software aspects of a system, and there was very little complex hardware logic, numerous adjunct documents were continually added to address the evolving hardware landscape. This historical DO-254 evolution is summarized in the following graphic:

Doc	Year	Basis	Themes
DO-178A	1985	DO-178	Processes, testing, components, four criticality levels, reviews, waterfall methodology
DO-178B	1992	DO-178A	Integration, transition criteria, diverse development methods, data (not documents), tools
DO-254	2000 –	DO-178B	Apply DO-178B type processes to hardware. No clear Objectives or criteria, more subjective/variable, covers all hardware.
AC 20-152 CAST-27 EASA SWCEH A(M)C 20-152A	2005 - 2019	DO-254	Clarify DO-254, require 178C-like processes, add HDL/VHDL logic focus, allowance for COTS HW, de-scope to avoid LRU/Circuit-Card-Assemblies when covered by ARP4754A like processes, etc.

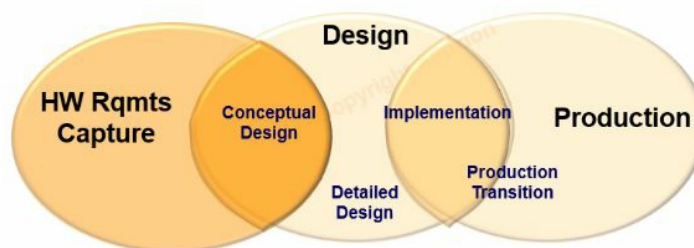
A common question on DO-254 projects is “Does hardware need both high-level and low-level requirements like DO-178C’s software?” The basic answer is “No – hardware only has one level of requirements, but has two levels of design: conceptual design then detailed design”. However, smart practitioners will understand the providing low-level requirements prior to implementing software reduces the number cause of software defects: assumptions. Similarly, hardware logic developers would be well-advised to similarly ensure their single level of hardware requirements was sufficiently detailed.

A common question when applying DO-254 to silicon-based logic is “Does DO-254 require logic reviews similar to DO-178C’s software code reviews?” The quick answer is “Yes”. While not explicitly described within DO-254 itself (since DO-254 preceded the mainstream utilization of HDL/VHDL in silicon/FPGA’s), such information was added to the DO-254 ecosystem via issue papers and advisory circulars (AC’s). The following figure depicts the necessary inputs and outputs to a hardware logic review. Note the ‘transitions’ from pre-to-post-logic review which are performed by engineers and audited by hardware Process Assurance.



The overall DO-254 Hardware Development Process is as depicted below:

DO-254 Hardware Development Process



Hardware Validation & Verification

As described in the preceding chapter, software is not validated in DO-178C since validation requires assurance of “completeness” which cannot be obtained until hardware/system testing of that software. While it’s easy to incorrectly intermix the terms “validation” and “verification”, they are completely different as depicted below:

- ✓ **Verification:**
“Does Implementation Meet Requirements?”
- ✓ **Validation:**
“Are the Requirements right?”

So what then really is validation for hardware? It’s actually quite similar to ARP4754A’s aircraft and system requirements validation. The primary methods for hardware validation are specified in the project’s Hardware Validation & Verification Plan and summarized below:

- * Validation Methods:
 - Bi-directional Traceability
 - Analysis
 - Modeling (see subsequent section)
 - Test
 - Similarity
 - Engineering Review
 - May require multiple methods from the above
- Validation must consider both Intended, and Unintended, Function

Since hardware’s DO-254 had the advantage of re-purposing software’s DO-178C, the verification principles of DO-178C largely apply to hardware: reviews of all artifacts associated with DO-254 compliance, tests of hardware implementation to requirements and assessment of logic coverage for DAL A/B, and additional analysis when the preceding combination of reviews and tests is not fully conclusive. DO-254’s verification regimen of Tests, Reviews, and Analysis is summarized in the following graphic:

Tests	Reviews	Analyses
• Hardware Requirements Based Testing • Logic Testing (A/B)	• Requirements • Conceptual & Detailed Design • Logic • Requirements Based Test Results • Low Level Test Results Review (A/B)	• Traceability Analysis (A/B/C) • Requirements Coverage Analysis (A/B/C) • Low Level Test Coverage Analysis (A/B) • Structural Coverage (A/B) • General System Analysis (A/B/C) • Static Timing • Device Usage • Thermal/Power • Pin Coverage Analysis (A/B/C)

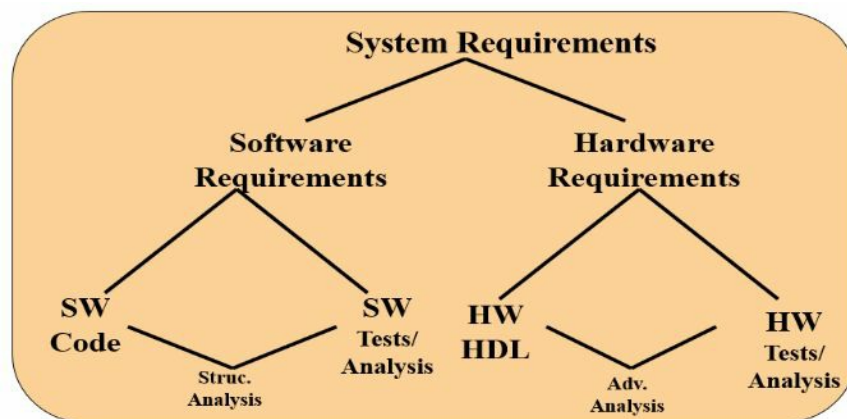
DO-254’s Verification Regimen of Tests, Reviews, & Analysis

The Hardware Validation & Verification Plan should cover the additional recommendations below:

- Specify exactly what you will **review**, when you will **review** it, by what criteria (reference checklist), and by whom
- Specify exactly what you will **test**, when you will **test** it, how will you **test** it, and by whom
- Specify what, if any, additional **analysis** will be performed
- Specify goals and objectives, rather than detailed “how to”
- Reference, not include, standards and checklists
- Address validation of derived requirements
- Describe different forms of verification and the environment for each
- Make it company generic: re-used on other projects with minor changes

Hardware versus Software Traceability

Another typical question regards traceability: “Does DO-254 need bi-directional traceability and is it similar to software?” Again, the short answer is “Yes”. The following graphic summarizes the hardware versus software traceability:



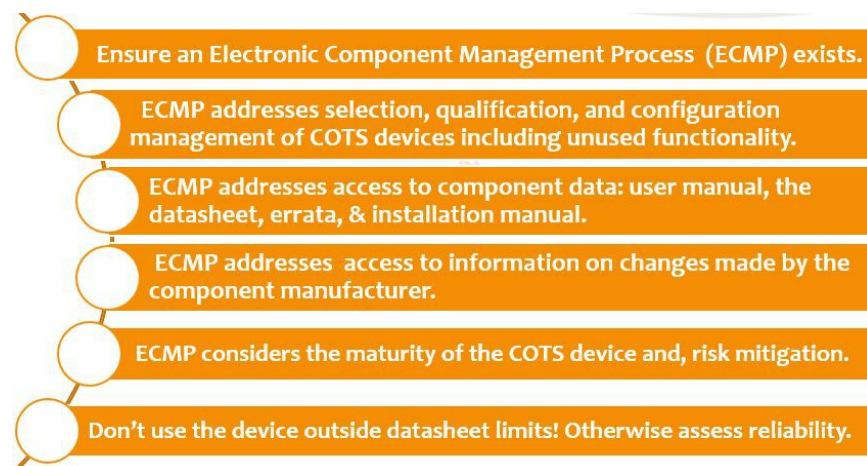
Hardware and Software Traceability

As denoted in the graphic above, traceability is always bi-directional and includes System level requirements, hardware/software requirements (but not

Conceptual Design or Architecture), tests, and implementation/logic. Hardware conceptual design, and software architecture, do not require explicit inclusion for traceability; however the tracing of requirements to logic “passes through” design; the key is that such design cannot obfuscate traceability.

COTS Hardware and IP Cores

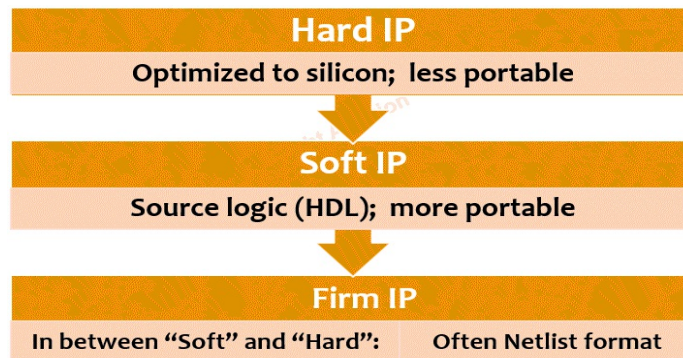
Whereas aviation software usage of Commercial Off-The-Shelf (COTS) software generally mandates full application of DO-178C, COTS hardware is more ubiquitous in aviation and more easily used in certifiable systems. Instead of redesigning hardware merely to comply with DO-254, the prevailing certification authority strategy is “Don’t introduce new errors via new hardware designs but rather retain proven electronic components while demonstrating good engineering management of that hardware. That good engineering management is demonstrated via an Electronic Component Management Process Plan (ECMP). The following figure summarize the ECMP and its usage:



COTS Hardware ECMP Summary & Usage

Another frequently asked DO-254 question pertains to IP Cores “Does DO-254 have special rules for IP Cores or can I simply use any IP Core?”. For this question, there is not a simple Yes/No, because IP Cores were largely non-existent in aviation at the time DO-254 was composed in the later 1990’s. However, IP Cores are increasingly prevalent and provide many advantages including reuse, quality, and time-to-market similar to software

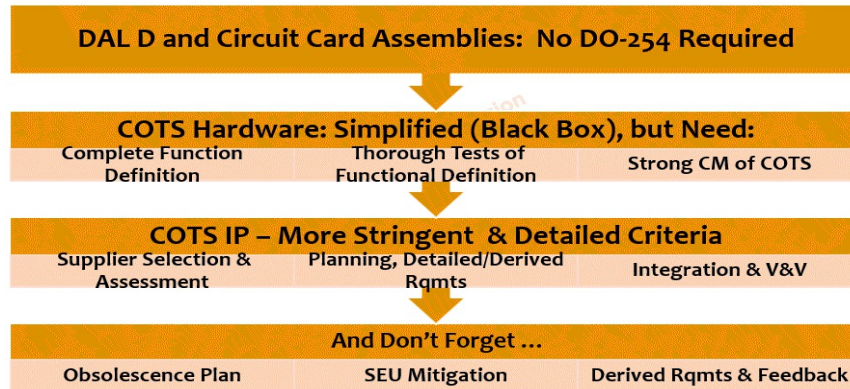
libraries and real-time operating systems (RTOS's). IP Cores are essentially categorized as Hard, Soft, and Firm as summarized in the following graphic:



Hardware IP Cores: Hard vs Soft vs Firm

While COTS IP Cores have generally undergone in-the-field debugging and the manufacturers have a vested interest in product quality, they still must be subjected to thorough requirements-based testing including robustness. For DAL's A and B, additional support from the manufacturer is required to ensure provable in-house development and configuration management processes are also applied.

Where software development evolution has included model-based development and object oriented technology (subjects dealt with in subsequent chapters of this book), basic software development usage of a high-order source language, compiler, and linker is rather staid, and little changed. Not so with hardware whose evolution has been extreme within aviation. Previous chapters of this book explained the rather small changes in software certification per DO-178A DO-178C. Obviously DO-254 could have benefited greatly from more frequent updates, or in fact a single update; such never occurred. Instead, the "application" of DO-254-based hardware certification was changed via more frequent ancillary certification memorandums such as AC 20-152, CAST-27, EASA CM-SWCEH-001 and AMC 20-152A, the latter of which was drafted in 2019 and approved in 2020. Corresponding key clarifications of these subsequent ancillary documents and AMC 20-152A are summarized in the following graphic:



Evolving DO-254 Clarifications including AMC 20-152A.

9

Aviation Requirements for Systems, Software & Hardware

By Vance Hilderman

Reviewed by:

- Mr. Marc GATTI, Thales AVS France SAS Scientific Director

Good requirements are the foundation of good products, and the only road to “great” products is through great requirements. In aerospace (or aeronautics), requirements are paramount for all aspects of products: hardware, software, systems, safety and now security. But how can one “prescribe” good requirements? What should an aerospace (or avionics) Requirements Standard contain? What are examples of weak, satisfactory, good, and finally great requirements? Are there differences between airborne and ground-based requirements or software and hardware requirements? Aerospace viability mandates we have the answers to these questions and so here they are...

First, please consider the following aviation requirements “quiz”. Do you know the answers? If you’re developing aviation systems, it is imperative to know the answers. These answers and many more are explained in this and the following chapters.

1. **T/F: On most aviation projects, most requirements pertain to Safety.**
2. **T/F: Aviation requirement development must follow a Waterfall.**
3. **T/F: The best way to assess aviation requirements is via test execution.**
4. **T/F: ARP47XX and DO-XXX provide clear guidance for developing and assessing requirements.**
5. **T/F: Most avionics defects are due to bugs and manufacturing defects.**
6. **T/F: When using model-based development, requirements must be**

textual and external to the model.

When asking certification authorities “Which is more important: requirements for airborne avionics systems, hardware, software, or ground based systems?” the truth is that the most valid and likely answer is “Yes”. Any other answer could lead the questioner to mistakenly believe they should prioritize one over the other. As concluded by many safety-critical development experts (and this author’s various books and papers), requirements are the foundation of aviation development. Worldwide safety-critical experts consistently state that the number one cause of safety-related defects can be traced to weak requirements. Such weak requirements imply reliance upon assumptions, with different engineers making different assumptions. Where assumptions are different, it follows that at least one of those assumptions is, or all of them are, **incorrect**.

However, all the aviation guidelines relied upon by aviation certification authorities are vague concerning methodologies for optimal requirements development. While many mistakenly believe this to be a weakness of these guidelines, the truths are three-fold:

1. *Aviation systems differ vastly in functionality, criticality, and chosen development methodologies; therefore no single requirements-writing guide would be sufficient.*
2. *Reliance is instead placed upon the requisite detailed “Requirements Standard” which each aviation project prepares, detailing that project’s specific approach to requirements development and criteria for assessing such requirements.*
3. *Gifted original designers have less need of detailed requirements than mere mortals. However, the vast majority of aviation systems are not “original” but rather continuing upgrades and variations of prior systems. Therefore, modifications are made not by the Originator but by less-informed engineers. And the verifiers, auditors, and certifiers of these systems never have the Originator’s level of product expertise. All of these non-originators strongly rely upon the Requirements to understand and assess the aviation product.*

A Word on “V&V” ...

Imagine a conversation between an all-too-typical aviation engineer and a non-engineer:

Non-Engineer:	<i>"What did you do today?"</i>
Engineer:	<i>"I did V&V."</i>
Non-Engineer:	<i>"What is V&V?"</i>
Engineer:	<i>"Verification and Validation. Or maybe Validation and Verification."</i>
Non-Engineer:	<i>"Did you do Validation, or did you do Verification?"</i>
Engineer:	<i>"Yes, I did V&V."</i>
Non-Engineer:	<i>"But did you do Validation, or did you do Verification?"</i>
Engineer:	<i>"Well, no one knows the difference so that's why we just call it 'V & V'!"</i>
Non-Engineer:	<i>"Engineers are strange ..."</i>
Engineer:	<i>"Yes."</i>

V&V is actually easy:

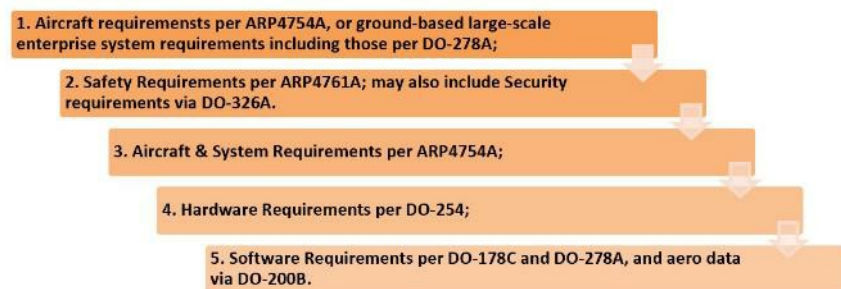
Verification: *does the implementation meet the requirements?*

Validation: *are the requirements 'right'?*

Now, ask the certification authorities which is more important: verification or validation. The politically correct answer is "Yes" again because verification should not have less importance than validation and vice versa. However, ask a truly experienced aviation engineer which is more important, and the only good answer should be: "Validation, because without good requirements it won't matter if you've verified them well!". That is exactly right, however what constitutes "good requirements"? Validation of requirements ensures they are correct and complete. For aviation software, validation is performed via hardware at the system level, therefore software requirements are technically validated via review (note that modeling via DO-331 greatly assists here by providing requirements visibility and validation via multiple levels from customer to system to hardware and software). For hardware and system requirements, however, validation includes review, simulation, modeling, bi-directional traceability, analysis, and any other combination of

requisite techniques to ensure completeness and correctness.

As aviation system complexity continually increases, a single level of requirements is insufficient. Perhaps early aviation could suffice with a single level of requirements but increasing complexity and larger engineering teams implies greater potential for mistaken assumptions. Remember from prior chapters herein: the number one cause of aviation system/software/hardware defects is “assumptions”. The best means to minimize assumptions is via improved requirements. Therefore, aviation systems have multiple levels of requirements including:



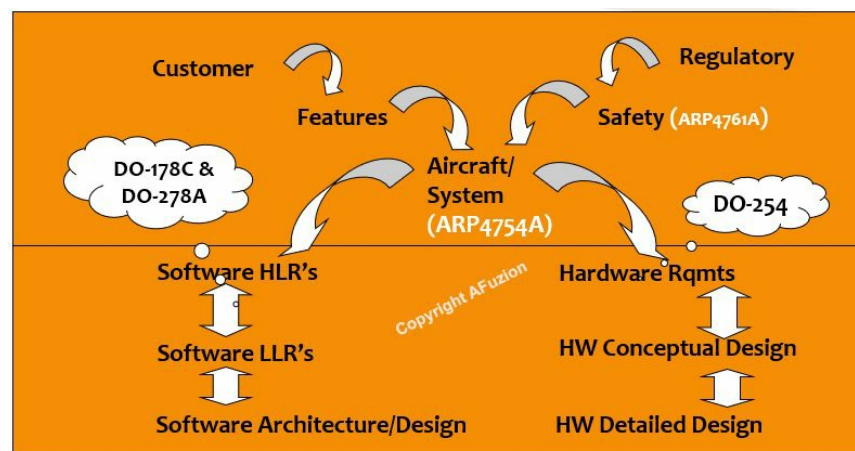
Typical Aviation Requirements Levels

Each of the above requirement domains likely represents successively increasing granularity, in the order depicted above. Throughout, additional Safety requirements can be decomposed or derived which further clarify necessary aspects of the system, hardware, and software. When systems are particularly complex, any one of the above requirement domains may be further subdivided into two or more levels of requirements. The end result is typified by multiple levels of requirements which enable higher quality through better understandability of the requirement relationships, and the ability to better validate, and then verify, those requirements. Aviation requirement development entails successively more detailed decomposition, with the requirements reviewed at each stage of refinement. For higher assurance levels associated with Hazardous or Catastrophic failure effects, requirement V&V must be proven to be independent, e.g. a different person or team following a process independent from the requirement developer.

In the case of software and hardware logic, it is increasingly common for complexity or software size to be so large that no single person understands more than a fraction of that logic. Therefore, airborne avionics (DO-178C) and ground/satellite based Communication Navigation Systems / Air Traffic

Management (CNS/ATM; DO-278A) software requires a default of two requirements levels: High-level Requirements (HLR's) and Low-level Requirements (LLR's); good hardware developers following DO-254 often apply two or more level of requirements as a “best-practice” even though such is not formally required within DO-254.

System requirements must embody the full system level (black-box) functionality including Safety requirements. System requirements are then decomposed and allocated to either hardware, software, or both. Software HLR's then state what the software does at the hardware/software interface level; HLR's should be devoid of software function/module level detail as that is instead specified via LLR's. It should be noted that a “best-practice” is to include these hardware/software interfaces (HIS) within a separate HIS document. The reason for two levels of requirements (HLR's then LLR's) is fundamental: while simple software functionality may be fully described via a single level of requirements, avionics software is complex and becoming more so. Therefore requirement elucidation, implementation, and verification are more reliably performed when specification is performed in two or more stages of refinement. The figure below depicts the aviation requirement ecosystem; note that “derived” requirements must also be included as such is necessary to address safety aspects (dissimilarity, health monitoring, etc.) and gap-filler requirements to ensure all functionality is addressed requirements and can thus be fully traced and verified. These derived requirements do not necessarily trace to a parent requirement, therefore require additional independent review for Safety.

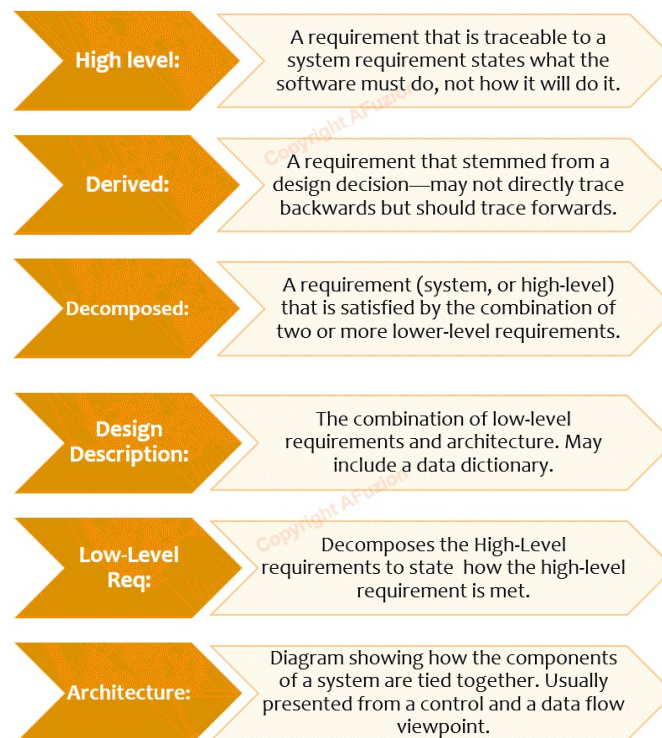


Aviation Requirements Ecosystem

In earlier versions of aviation guidelines, such as DO-278 and DO-178B, the formal objectives associated with HLR and LLR requirements were particularly vague; there was seemingly no formal mandate to “go back and improve requirements”. DO-178C and DO-278A improved that by mandating that LLR’s be quite detailed down to logic branches and forcing requirement improvement (usually via LLR clarification) when structural coverage analysis indicated uncovered code structures due to weak LLR’s. Therefore legacy systems, particularly ground-based CNS/ATM systems, have been commonly noted by this author and his employees to contain rather weak LLR’s.

Requirement Terminology:

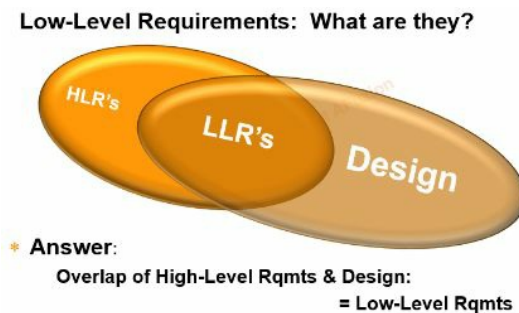
Aviation requirements have a unique taxonomy, somewhat different than non-aviation industries. The intricacies of key aviation requirement terminology and application are detailed in the following paragraphs with the figure below summarizing:



Aviation Requirements Terminology

The following diagram introduced in a preceding chapter is again shown below to emphasize the relationship between software requirement stages and

design:

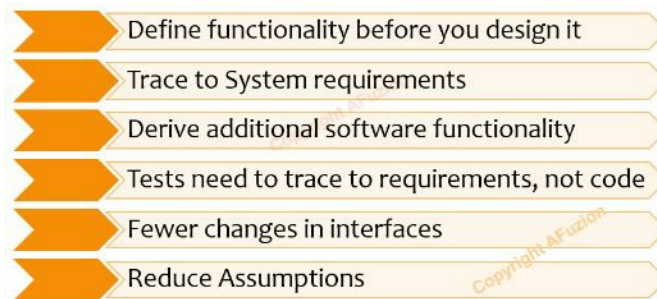


Aviation guidelines do not provide a recipe for composing requirements, nor do they define a distinct boundary between HLR's and LLR's, or between LLR's and Design. Instead, each project is responsible for providing such a verifiable “recipe” via a formal Requirement Standard by which the requirements will be evaluated. However, a best-practice is to first fully define the requirements (high-level, and low-level as appropriate) prior to design. Regardless, the design must be reviewed by considering all related requirements, both high-level and low-level. Remember: all requirements are subjected to Test, whereas design is not tested but rather documented and reviewed. Since all requirements are subjected to testing regardless of their designation as high-level or low-level requirement, the actual distinct boundary between High and Low is less important. Now, if you have perfect decomposition of high-level requirements to low-level requirements, and you fully test the low-level requirements, must you still test the high-level requirements? The answer is “Yes, since both high-level and low-level requirements are inputs to the test review process, you must consider both high and low-level requirements when reviewing the test. In real-life, there are no such thing as “perfect” requirements and test cases should be developed prior to designing and implementing the software so that that consideration of test cases can be used during the requirement review process to improve those very requirements.

High-Level Requirements.

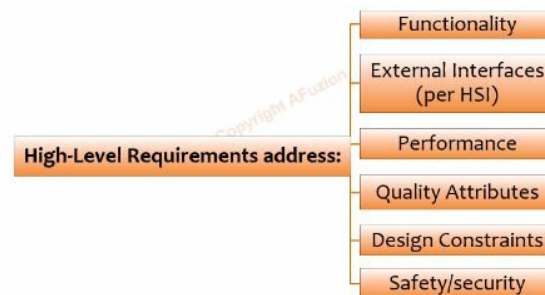
HLR's development follows System requirement definition but precedes LLR development; HLR's must be formally reviewed (see below) prior to development of the associated LLR's. The following graphic provides the key reasons why HLR's are necessary for any aviation system potentially

having a safety and security impact and thus requiring software and hardware certification:



Key Reasons to Develop High-Level Requirements

HLR's are the "bridge" between System requirements and LLR's. Good HLR's should address and elucidate the following software aspects:

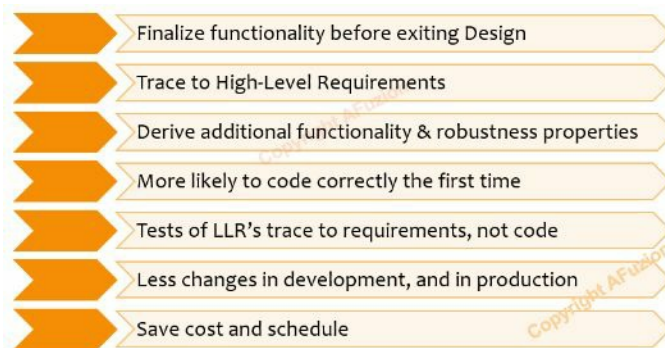


Attributes Addressed by High-Level Requirements

HLR's (and LLR's below) which are ascertained via analysis (or decomposition) of System Requirements or System Architecture are termed "non-derived" because of that direct system relationship. HLR's which come from Safety-Related Requirements are usually called non-derived but the derived/non-derived designation is less relevant because that HLR inherits the "safety" attribute from its safety source, so it must be fed back to the Safety process (this is the real (ARP4754A and DO-178C, but not the former DO-178B) reason for "derived" and "safety" requirements: all such must be independently reviewed by Safety as part of that formal Safety Feedback process (which must also be audited by Software Quality Assurance and by Systems Process Assurance). HLR's which come from analysis of Safety Assessments (as opposed to analysis of Safety Requirements) are ALWAYS "Derived" requirements (no parent) and also must have the Safety attribute set for requirements management.

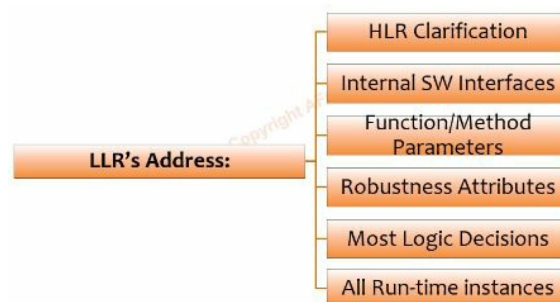
Low-Level Requirements.

LLR's follow HLR requirement definition and precede or accompany design; LLR's must be formally reviewed (see below) prior to development of the associated logic. The following graphic summarizes why LLR's are necessary for any aviation software potentially having a safety and/or security impact and thus requiring software and hardware certification:



Key Reasons to Develop Low-Level Requirements

LLR's are the "bridge" between HLR's requirements and Design/Implementation. Good LLR's should elucidate the following software aspects:

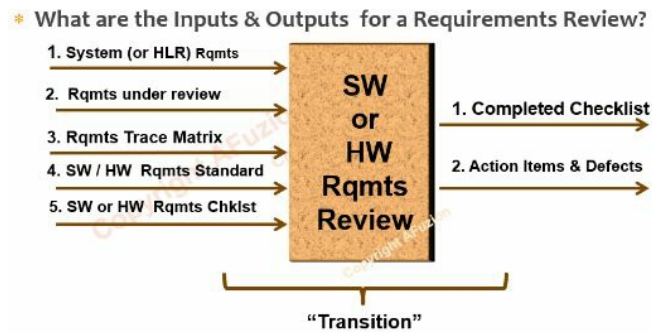


Attributes Addressed by Low-Level Requirements

Aviation: Fix the Requirements or Fix the Product?

A key facet of aviation requirements is early validation. Validation assesses the correctness and completeness of requirements. Since validation requires hardware considerations, requirements need to be validated inclusive of System and Hardware considerations. All requirements must be verified, with verification consisting of formal Reviews and later Testing; where Testing is not conclusive, additional Analysis may be required. Obviously, it is better to improve requirements prior to implementation as defect prevention is more

cost-effective than defect correction later. Verification, on the other hand, asks the question, “Does the implementation meet the Requirement(s)?” All requirements must be formally reviewed, where “formal” means provable use of review criteria along with record-keeping and defect correction. For example, an aviation software or hardware requirements review would have inputs and outputs which look like this:



As depicted above, there are five inputs to a formal requirement review; all five must be under configuration control and proven to be used to perform the actual review. These five inputs (which include derived requirements if applicable) to a formal requirement review comprise the *entry* criteria, whereas the completed requirements review checklist and action item/defect records comprise the *exit* criteria. This movement from activity entry to exit comprises a “transition”. The requirements verifier performs the transition and quality assurance audits the transition. For reviews, the number of reviewers is unimportant (in fact, it is this author’s experience that the best reviews are accomplished when fewer but better reviewers are used; team size in complex systems is **generally inversely proportional** to resultant quality and most certainly productivity.) The key to the requirements review is the application of the corresponding Standard and as well as the Checklist. Typical high-quality safety-critical requirements standards are detailed and 20+ pages in length; high-quality requirements review checklists are similarly detailed and 6-8+ pages in length. This contrasts sharply with non-safety-critical products which often lack requirements standards and checklists, or, when present, are still very light.

In the earlier aviation guidelines, requirement necessity and quality was recognized but less enforced. Today’s guidelines require continual requirement refinement such that both defects and logic testing coverage

shortfalls entail a primary remedy of improving requirement clarity. DO-178C and DO-278A went even further and require tests to affirm software code coverage (“structural coverage”) to be based upon requirements. This means LLR’s must be very detailed and provide sufficient detail to be used in code reviews for the logic decisions.

Derived Requirements: English versus “Avilish”

Years ago, this author held a holiday party for over one hundred of his international aviation engineer employees. His spouse noted the engineers spoke a unique blend of English which was heavily influenced by aviation terminology, much of which she could not understand. They surmised that even the native English speaking engineers really didn’t speak English but rather “Avilish” (rather *Aviation English*), which became the company’s slang for “Aviation English”. In Avilish, normal English terms took on an entirely different meaning. Perhaps the most significant transformation of aviation English is the usage of the term “derived requirement”. Now, in proper English and even American English, the term “derived” means “to come from”. For example, “the human’s DNA is derived from its parents”, and “an optimal travel route can be derived from examining a map”. So much for regular English ...

In aviation, the term “derived requirement” is defined as “a requirement derived from a safety assessment process or engineering decision to mandate particular functionality or capability which may be unrelated to any parent requirements and to close functionality gaps not otherwise decomposable from parent requirements.” Safety-critical systems almost always have safety-related aspects which directly or indirectly improve safety: redundancy, built-in test, health monitoring, etc. all comprise safety-related functionality which may be devoid of parent requirements mandating such specific safety aspects. The simple analogy is your office building: quite likely, your office has smoke detectors, sprinkler systems, and a secondary door; these items do indeed perform their respective functions, but they are safety related and therefore not functionally required for use of the office. That smoke detector, sprinkler system, and secondary door will hopefully never be used; but like the requirement to have a seatbelt and airbag in your car (also Safety requirements), they are derived from a safety assessment or

to meet specified safety requirements and therefore are termed “derived” for aviation. Therefore, in aviation these aspects are deemed to be specified via derived requirements per the aforementioned definition. Examples of aviation derived requirement aspects include the following:

- Safety requirements such as health monitoring for reset, deactivation, or switchover
- System sampling rates
- Redundancy
- Fault logging/reporting
- Architectural choices: triplex versus dual-monitored
- Watchdog timer strobe frequency
- Dissimilarity
- Definition of hardware/software inputs and outputs.

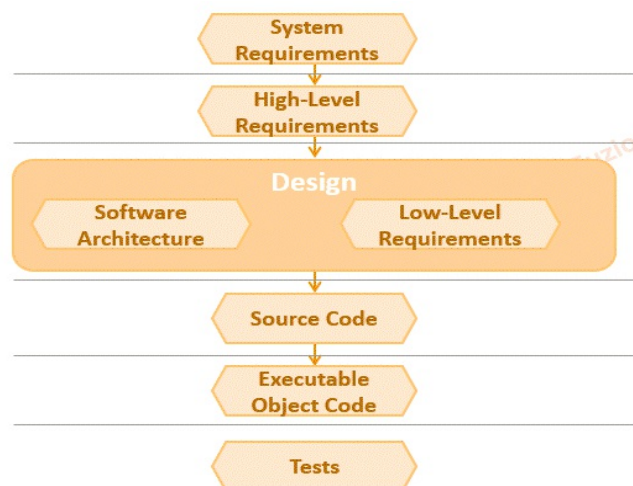
Now, you could call all the above “Functional Requirements” by simply adding parent requirements for such. But since Safety related requirements must be shown to have been fed back to the Safety process for official (and documented) Safety review, such designation of Safety requirements as instead Functional requirements could potentially bypass the requisite Safety review. So don’t do that. Instead, call the baby horse a foal, not a horse, and give it the proper nurturing and review attention it deserves: “Derived (safety) Requirement”.

Requirements = “What”, Design = “How”

Everyone knows, or should know, that requirements state what measurable functionality will be performed, whereas design states how the functionality is then implemented. However, consider a designer versus an implementer: designers improve their designs by prototyping, and prototyping is implementation. When the implementer changes implementation, they often alter the design. Therein lies the dichotomy: the boundary between requirements and design is indeed vague, and such vagueness implies subjectivity which belies assumptions. This is particularly true in aviation:

when is an aircraft requirement a system requirement is a software/hardware requirement is a design decision? Whew. Let's proceed ...

In the case of aviation software, both airborne and ground-based, the associated certification guidelines DO-178C and DO-278A, respectively, require high-level requirement development to precede high-level requirement development. High-level requirements should always be devoid of design detail. However, low-level requirements are considered to be tangential to, and a part of, software design which includes both architecture and low-level requirements. The overall aviation software engineering phase progression is depicted below:



Aviation Software Engineering Phase Progression

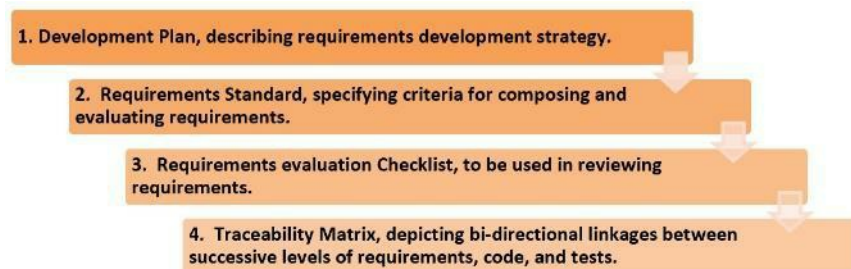
Making Good Requirements & Requirements Standards

Have you ever watched professional billiards players? To a casual observer, they seem skilled but also often lucky. Not only do they rarely miss, but the cue-ball lands in a perfect position for the next shot. Good amateurs make the shot, while professionals actually plan several shots ahead and win the game by running the table. This is the difference between professionals versus amateurs. In aviation, luck is rarely involved except for bad luck when you read about the crash the next day. Most aviation accidents are not caused by pilots but rather the pilots' inability to mitigate. However, that is a topic addressed separately. Just as professionals plan several moves ahead, aviation engineers must do the same when developing requirements. Aviation requirements should be based upon the desired end-result. Instead of making

the amateur mistake of trying to think of textual requirements one-by-one (as in amateur billiards), the professional should envision the end-game, e.g. what exactly comprises a successful aviation system with all its interactions.

The aviation guidelines provide precious little information on how to develop requirements, as such development techniques are highly subjective and thus difficult to assess. Guidelines such as DO-178C (airborne software) and DO-278A (ground/satellite based systems and software for Communication and Navigation plus Air Traffic Management – CNS/ATM) instead cite “Objectives”, including objectives for assessing requirements. They are called Objectives for the simple reason that the criteria for assessing Objectives can be unambiguously applied. However, good requirements also have subjective criteria which are not mandated or even addressed within the aviation guidelines; such are subjective and thus not assessable by the generic Objectives within the guidelines. How then do aviation certification authorities ensure inclusion of such subjective requirements assessment criteria? Simple: that’s the purpose of the mandated Requirements Standard mandatory for DO-178C, DO-254, and DO-278A (and highly recommended for ARP4754A). Aviation guidelines call for the developer to create and apply a mandatory Requirements Standard for the higher assurance levels, e.g. those levels where a system failure could cause injury or worse.

Now, back to the end-game: developing successful aviation systems. While the aviation certification guidelines do not require any particular methodology or approach to developing requirements, the developer is required to prove compliance. Therefore, the aviation certification guidelines “require” the following artifacts to be present prior to developing requirements:



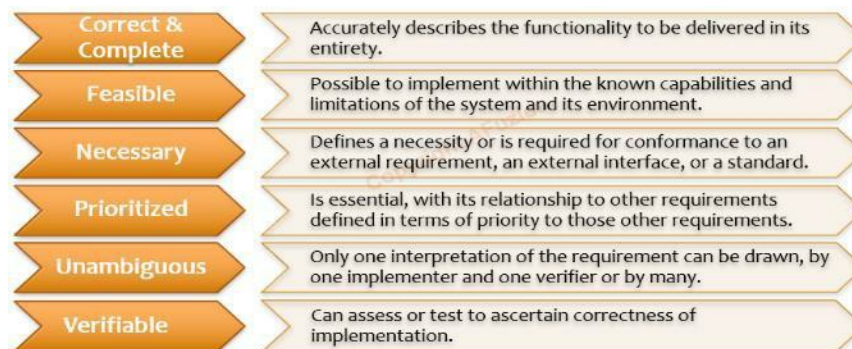
Necessary Inputs to Start Compliant Aviation Requirements Development

The above four inputs to the aviation requirements development process need

to be defined prior to initiating requirements development and the subsequent evaluation of those requirements needs to use all four inputs to ensure guideline-compliant processes were followed. What separates the amateurs from the professionals within the game of requirements development? Efficiency, accuracy, and quality:

- **Efficiency:** *elucidate the requirements early and with minimal iterations;*
- **Accuracy:** *ensure requirements are correct and complete;*
- **Quality:** *ensure requirements are compliant meeting all transition (input) criteria, with review and audit records.*

To accomplish requirement development efficiency, accuracy, and quality, it is necessary to think like professionals and consider the end-game first. For amateurs, the goal of aviation requirements is to satisfy customer expectations via meeting customer requirements. For professionals, the end-game or “goal” of successful aviation systems is more encompassing: comply with all certification guidelines, satisfy safety and then functional requirements including those of the customer, interface cleanly with other systems, and promote reusability. In other words, the professional requirements developer is planning several moves in advance and considering much more than the next move of simply meeting customer requirements. What then are the attributes of professional aviation requirements? The following figure summarizes; note that every effort should be made to promote reusability of existing technologies – aviation is already complex and need not be made more so:



Attributes of Good Aviation Requirements

Weak Requirements

Weak requirements abound everywhere, but typically not for the assumed reason of incompetent or unknowledgeable engineers. Instead, weak requirements are commonly the result of the opposite condition: smart engineers believing the requirements to be for their or other equally smart persons' benefit or weakly expressed goals between themselves and managers/customers. Therein lies the problem: the truth is that the other engineer's needing to read the requirements in order to verify, or in the future change, the stated functionality most certainly do not have the knowledge of the original creator or implementer. If systems are relatively small or engineered solely by exceptionally brilliant engineers (think NASA and the early space program), aviation requirements are, frankly, less important. As the systems become larger and more complex, a greater number of engineers are required which drives the average skill level closer to the definition of "average". Those are the engineers whose systems suffer the most from weak requirements. Truth be told, the vast majority of aviation development is built upon evolutionary progression of prior aircraft and systems; it's quite rare to have an all-new aircraft or system. Thus rarely are the original developers of aircraft and systems fully involved in the derivatives so weak requirements lose this necessary knowledge of the originators.

Consider the following obviously weak requirement:

"The System shall, as a goal, calculate aircraft heading, altitude, and global position to a sufficient accuracy."

Obviously, the preceding requirement is weak. However, if we examine this requirement word by word, we realize that requirement is actually *very* weak.

- ***"The System"***: Insufficient. Which system? What version and configuration?
- ***"as a goal,"***: Insufficient. Goals are not requirements and cannot be assessed.
- ***"calculate"***: Insufficient. Calculate how? What Algorithm? When? How often?
- ***"aircraft heading"***: Insufficient. True heading or magnetic? Accuracy?
- ***"altitude"***: Insufficient. Above sea level or terrain? Feet or meters?

Resolution?

- ***“and global position”***: Insufficient. Lat/Long or GEOREF? Resolution?
- ***“to a sufficient accuracy”***: Insufficient. Entirely unclear as to tolerance.

In fact, there is only one “good” word in the aforementioned requirement: the word ***“shall”***. Keep it and fix everything else.

Requirements versus Goals & Subjective Desires

A common problem with safety-critical systems is stating system goals as if they were requirements. Goals, like New Year resolutions, are wonderful ideas and everyone would benefit from having them. However, goals cannot be objectively assessed and guaranteed. The following are common aviation goals that are all too often expressed in requirement format:

- ***“The aircraft must be safe.”***
- ***“Serious accidents cannot occur.”***
- ***“Passengers & crew can never be killed or injured.”***

The above are truly great goals, but they are not requirements. Another problem often seen in requirements is expressing them via subjective desires. Humans have infused subjective desires into their communications since the beginning of the written word. Distances (“far”, “short”), size (“large”, “small”), and temperature (“hot”, “cold”) were phrases recorded in early human languages. But what exactly is their meaning? The reader can imagine a general idea but nothing discreetly verifiable. Consider the following subjective words:

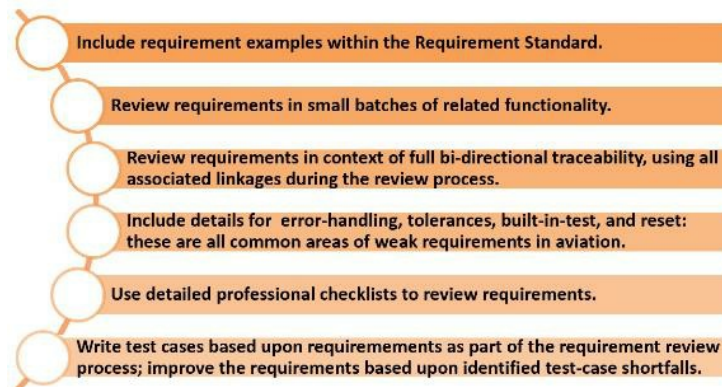
- *Immediately*
- *Respond*
- *Short*
- *Simultaneous*
- *Modular*
- *Nominal*

The above words are commonly found in safety-critical requirements but should be replaced with objective terms. The following requirements were taken from aviation requirements specifications received by this author in his

prior life as Co-founder/CEO of the world's largest aviation certification services company; they are not objective requirements and, in fact, are barely even weak goals:

- *The software shall be robust.*
- *The hardware shall degrade gracefully under stress.*
- *The software shall be developed in accordance with modern programming practices.*
- *The system shall provide the necessary processing under all modes of operations.*
- *Computer memory utilization shall be minimized to accommodate future growth.*
- *The software and hardware shall be easy to use.*

Clearly, developing great requirements entails a unique mixture of science, art, skill, and practice and knowledge about the global system functionalities and capabilities (don't promise the moon expect for your fiancée). However, it also requires good Plans, Standards, Checklists, and methodologies. The following best practices are not required by the aviation guidelines as they cannot be objectively assessed, but they should be embodied within the requirement engineering lifecycle:



Requirement Development Best Practices

While the above best practices are not required, anyone developing aviation requirements without incorporating them is relying upon luck to succeed. Amateurs hope for luck; professionals plan for success. We can all be Professional.

DO-200B Aeronautical Data

By Vance Hilderman

Reviewed by:

- Mr. Don Coleman, FAA DER
- Mr. Tom Roth, FAA DER

DO-200B, “***Standards for Processing Aeronautical Data***,” is a cornerstone within modern aviation. While DO-178 and DO-278 garner a greater portion of mindshare, DO-200B is the workhorse upon which all modern aircraft rely, both directly and indirectly. Why? Because DO-200B governs the means by which data necessary for safe aircraft operations is prepared, updated, utilized, and maintained. It is of no minor significance that DO-178 and DO-278 are explicitly termed guidelines as the term “Considerations”, not “Standard”, appears in each of their titles; conversely DO-200B is actually a minimum standard as stated in its title.

Consider the following statements and assess whether they are true or false; answers and explanations are provided within this paper:

1. ***T / F: DO-200B applies to Terrain Data, Navigation Data and Engine Data.***
2. ***T / F: DO-200B is a major rewrite of DO-200A.***
3. ***T / F: The three basic DO-200B processes are Data Quality Requirements, Data Processing Requirements, and Quality***
4. ***Management.***
5. ***T / F: The Supplier is primarily responsible for ensuring data usage integrity.***
6. ***T / F: A Type 1 Letter of Acceptance requires testing on the specified avionics end-item.***
7. ***T / F: All Data Processing tools need to be qualified for Level 1 data.***

If the above questions were truly easy, congratulate yourself on your genuine expertise. If they simply seemed easy, then the information in this chapter that follows is for you. In fact, answering the above without understanding

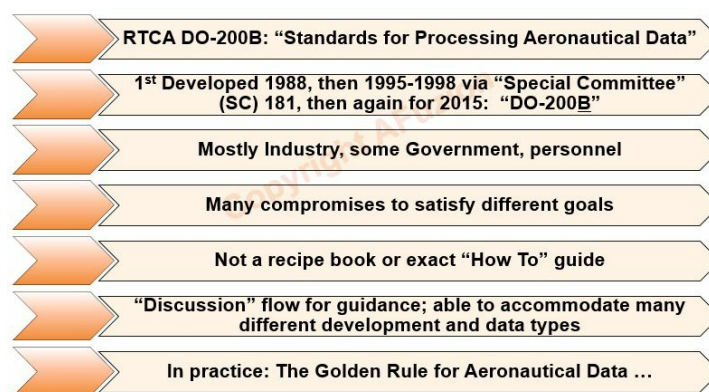
the overall data chain and quality assurance role in DO-200B is like understanding Fourier transforms without first understanding The Calculus: impossible for mere mortals ...

To succinctly summarize, DO-200B provides guidance for the following aeronautical aspects:

- Minimum standards and guidance for processing aeronautical data;
- “Aeronautical Data” = data used for navigation, flight planning, terrain awareness, flight simulators, etc.;
- Criteria for developing, changing, and supporting aeronautical data;
- Ultimately providing the user with assurance of data quality.

Aeronautical Definition:

First, what is the meaning of the term “aeronautical”? “Aeronautical” applies to more than just aircraft, air traffic control, training aids, and navigation. The term “aeronautical” was chosen accordingly because it is a superset of “aircraft.” Whereas DO-178 and DO-254 are intended for airborne software, DO-200B applies to data which may never be present in an aircraft but in some way influences aviation-related safety. This includes aircraft operations, simulation, training, planning, etc. Hence the term “Aeronautical.” A brief synopsis of DO-200B is shown below:

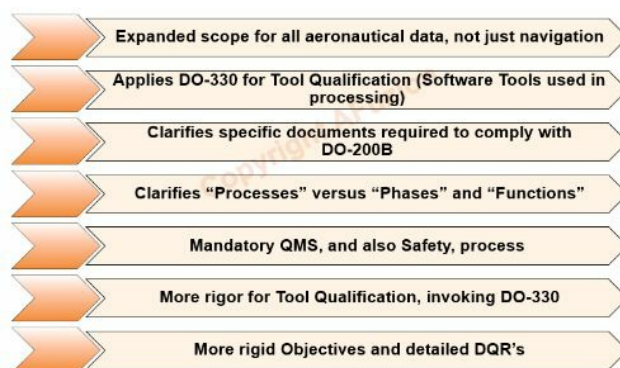


DO-200B Brief Synopsis

Minimum Standards:

DO-200B is a modest upgrade to DO-200A, with increased focus on the

overall aeronautical data (200A was somewhat more “navigation” oriented), data security, aeronautical data chain, increased scope/definition of tool qualification and DO-330 and expanded definitions/clarity. DO-200B provides the “minimum” standards. The user is encouraged to, and often must, do more than the “minimum” guidance provided within DO-200B. Since aeronautical data can take on so many different forms, and the future of aviation will assuredly include data forms unknown today, DO-200B cannot possibly include sufficient details for each data type. A similar situation exists for software and hardware via DO-178 and DO-254: those standards apply to virtually all airborne avionics, from bathroom lights to thrust reversers to navigation systems; the added requirements for each system type are not addressed within them. However, for airborne avionics additional system-specific requirements are contained within other required certification documents such as Technical Standard Orders (TSO’s – please see the preceding TSO chapter for details). Unlike those airborne systems, additional data-specific aeronautical data requirements are largely unaddressed within supporting documents, with the notable exception of Advisory Circular (AC) 20-153B, “**Acceptance of Aeronautical Data Processes and Associated Databases**” which is freely available and is a must-read for all DO-200B practitioners. It is therefore imperative for users of DO-200B to remember the “minimum” standards are almost certainly insufficient for most projects: it is incumbent on the user to further elucidate additional standards criteria specific to their data and processes. The following figure summarizes the key differences between DO-200A and DO-200B:



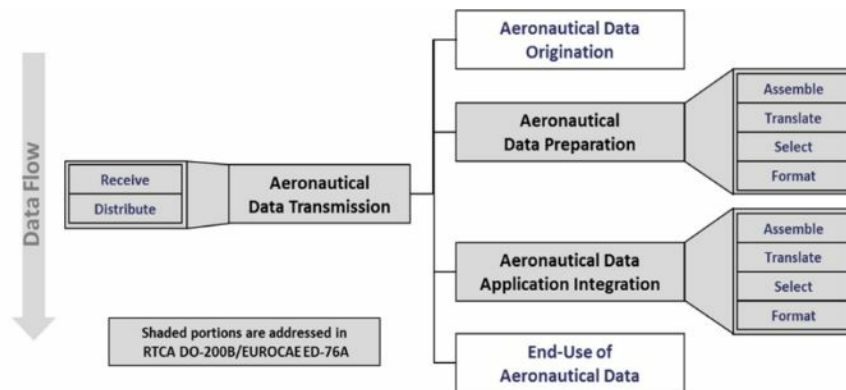
Key DO-200A versus DO-200B Differences

DO-200B’s provides “recommended standards” as opposed to strict requirements. The document was developed by 245+ persons from around the

world, principally from aviation product development companies but also inclusive of key certification authorities and military personnel. As with nearly all committee-based standards committees, DO-200B's authors had to reconcile conflicting interests with an all-too-common desire to develop the impossibly perfect document, e.g. "all things to all people". Thus DO-200B's 77 pages apply to both large and small aeronautical data sets, different criticality levels, and different users. The prevailing concern is that "provable quality systems" outweigh "strict process steps" where aeronautical data is concerned. Each user must carefully consider then analyze their contribution to safety by asking the following questions for each step within their data chain:

- *"Could their data usage fail to detect an error?"*
- *"Could their data usage insert an error?"*
- *"Could their data usage propagate an error?"*
- *"What are answers to the above questions considering the data development and usage ecosystem and tool-chain?"*

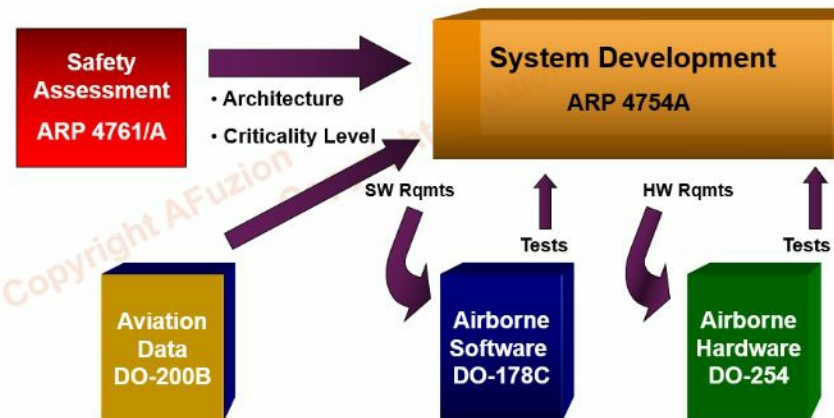
The scope of DO-200B and the aeronautical data chain process is depicted in the following graphic:



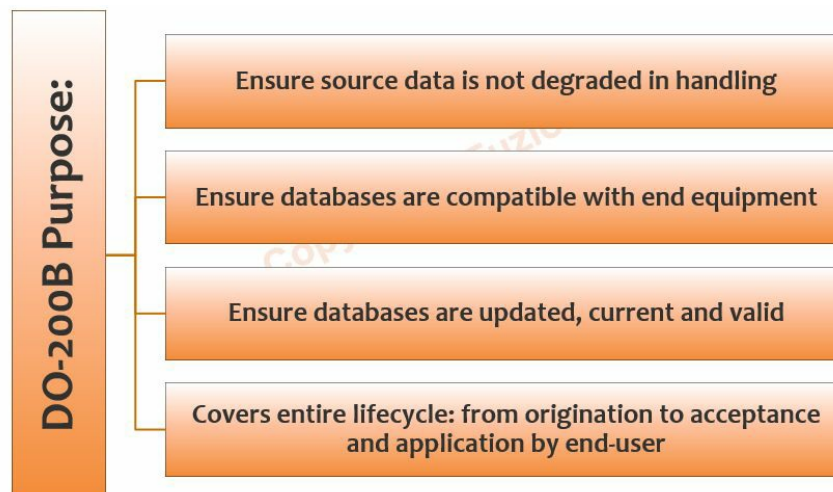
Shaded blocks addressed by DO-200B/EUROCAE ED-78

Aeronautical Data & The Aviation Ecosystem

DO-200B is part of the overall Aviation Development Ecosystem (a term coined by AFuzion), which includes airborne and ground-based software/systems combined with formalized Safety processes. The following graphic depicts these relationships:



The top four purposes of DO-200B are summarized in the following graphic:



DO-200B Top Four Purposes

DO-200B requires planning, data requirements, processing requirements and proof of related processes; these items must then be validated and verified throughout the data-chain, beginning with data receipt and ending with data transmission. Both data Supplier and User share responsibility for ensuring data integrity. Required documents for a typical DO-200B activity are depicted below:



Typically Required DO-200B Activity Documents

The primary DO-200B plan is the Compliance Plan with this plan covering the topics shown in the following graphic:



DO-200B Compliance Plan Key Topics

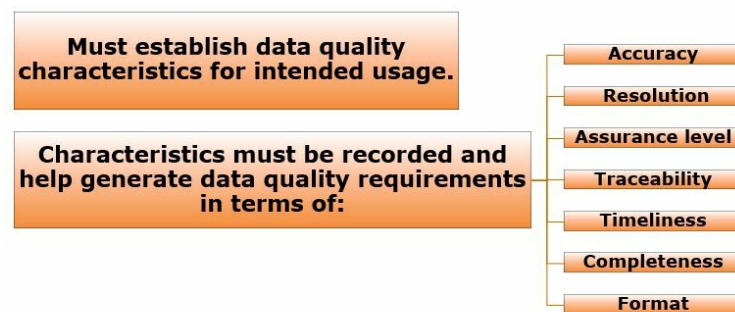
Aeronautical Data Definitions.

This aviation world already has its own vocabulary, with the slang word “Avilish” previously introduced in a prior Chapter herein to mean “Aviation English”. DO-200B takes that one step further and introduces a number of key terms which have little contextual meaning outside the realm of aeronautical data as follows:

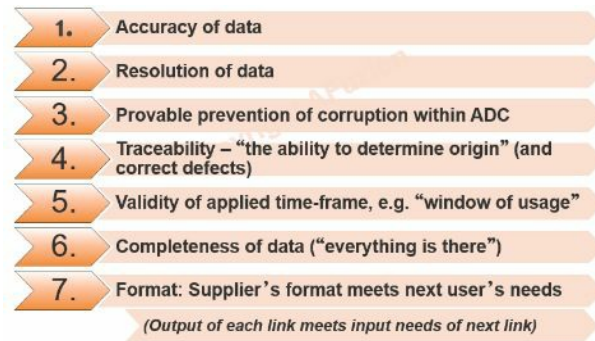
- ❑ **Aeronautical Data:** Data used for aeronautical applications such as navigation, flight planning, flight simulators, terrain awareness and other purposes, which comprises navigation data and terrain and obstacle data.
- ❑ **Aeronautical Database:** An Aeronautical Database is data that is stored electronically in a system that supports airborne or ground based aeronautical applications. An Aeronautical Database may be updated at regular intervals.
- ❑ **Assemble:** The process of merging or compiling aeronautical data, sometimes from multiple data suppliers, into a database and establishing a baseline for subsequent processing. The assemble phase includes checking the data and ensuring that detected errors and omissions are rectified.
- ❑ **Assurance Level-** The degree of confidence that a data element is not corrupted while stored or in transit. This can be categorized into three levels: 1,2, and 3; with 1 being the highest degree of confidence and 3 the lowest.
- ❑ **Correct Data:** Data meeting stated Quality requirements.
- ❑ **Data Quality:** A degree or level of confidence that the data provided meet the requirements of the user. These requirements include levels of accuracy, resolution, assurance level, traceability, timeliness, completeness, and format.
- ❑ **End-User:** The last user in an Aeronautical Data Chain. Aeronautical data end-users are typically aircraft operators, airline planning departments, air traffic service providers, flight simulation providers, airframe manufacturers, systems integrators, and regulatory authorities.
Usually responsible for data quality and DO-200B compliance!
- ❑ **Error:** Defective or degraded data elements or lost or misplaced data elements or data elements not meeting stated quality requirements.
- ❑ **Originator:** The first organization in an Aeronautical Data Chain that accepts responsibility for the data. For example, a State or RTCA DO-200B / EUROCAE ED-76-compliant organization.

Aeronautical Data Quality Attributes

Aeronautical data has the following quality attributes, generally applicable for each data item or set:



The Data Quality Attributes are summarized in the following graphic:



Data Accuracy and Relationship to DO-201:

Prior to applying DO-200B, it is important to understand applicable data requirements and data origination; that is the purpose of DO-201, “*Industry Requirements for Aeronautical Information*”. DO-201 ensures quantified data accuracy, resolution, criticality, calculation, and procedures. Data accuracy requirements must be specified when applicable. For example, consider a VOR. Each VOR has a unique identify; that identity is simply a “name” and the name must be exact (correlated to an actual VOR with the same name). There are no accuracy requirements for a name. However, the same VOR has a location and a height; both location and height have corresponding accuracy requirements and it is those accuracy requirements which must be known, measurable, and assessed.

Resolution of Data:

Aeronautical data often has associated resolution attributes. The required data must be specified, and it is based upon the intended use of that data. Often different preparers of aeronautical data do not fully understand the intended use of the data, so it is imperative that intended use be documented and known. Resolution only applies to items which can be measured such as aeronautical data. For example, in the preceding VOR example, the VOR name is an identifier and does not have a resolution attribute, whereas the location/height attributes are measurable, and those measurements must have documented resolution characteristics. Data resolution is defined (incorporated) into format of output data, per DO-201. However, if the particular data item is not covered in DO-201, then the user must perform a safety assessment on that data to determine appropriate resolution characteristics and document them.

Assurance Levels versus Criticality Levels:

Whereas DO-178, DO-254, et al. use associated criticality levels, DO-200B uses Assurance Levels 1, 2, or 3. The relationship between assurance levels and criticality levels is depicted below for Part-25 (larger) fixed-wing aircraft:

Remember:

Level A	Level B	Level C	Level D	Level E
$\leq 1E-09$	$\leq 1E-07$	$\leq 1E-05$	$> 1E-05$	NA

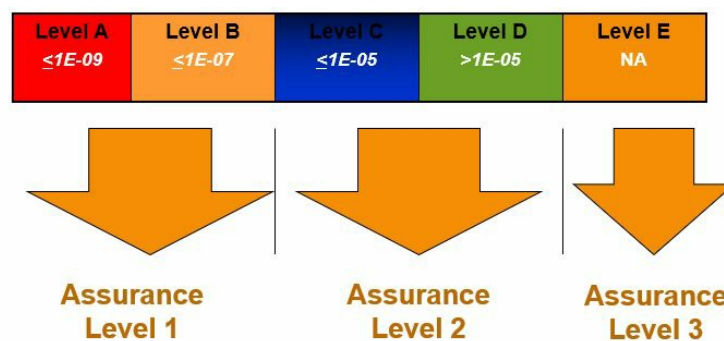
And:

Table B-1 Assurance Levels

Data Process Assurance Level	Related Requirement on State-Provided Data (ICAO)
1	Critical
2	Essential
3	Routine

DPAL	Probability of undetected error
1	$\leq 10^{-9}$
2	$\leq 10^{-5}$
3	Not applicable.

DO-200B's three assurance levels mean there are three different categories of data, with Level 1 being the most stringent. By using different assurance levels, less critical data items can undergo reduced certification aspects; thus decreasing the time and budget associated with processing these less critical items. For an additional view of assurance levels within DO-200B and how they compare to aviation safety, consider the following equivalency:

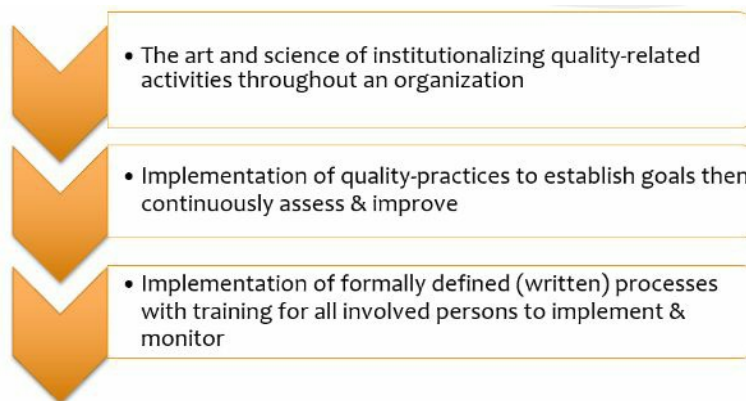


DO-200B's compliance planning process must ensure key data integrity assurance activities are planned, performed, assessed, and recorded. To accomplish such DO-200B data compliance, the following six integral activities are necessary to be applied throughout all data development and deployment activities:



DO-200B's Quality Management System

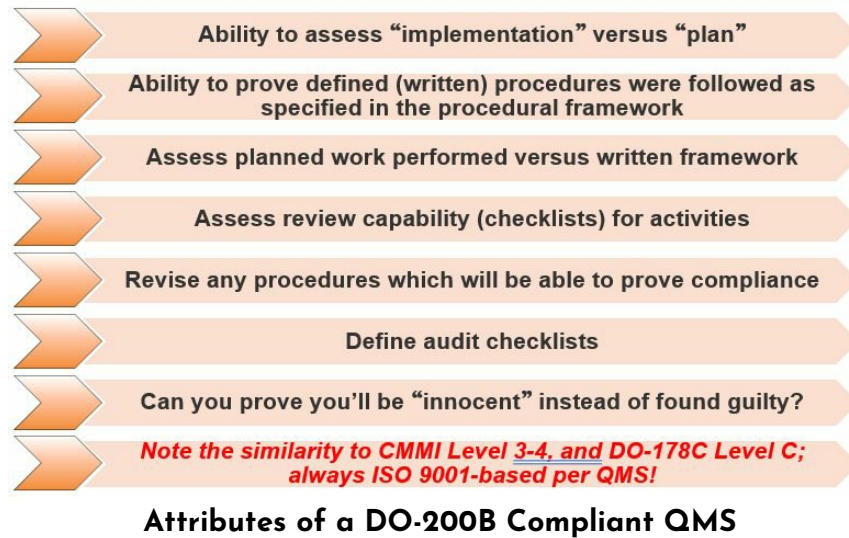
Whereas real-time aviation software and hardware system are carefully developed with safety-related aspects individually assessed, aeronautical data processing is almost always built around commercial off-the-shelf (COTS) hardware and software for data processing. Since the development of such COTS hardware and software cannot be subjected to aviation-grade internal scrutiny, aeronautical data compliance instead focuses upon the data processing framework developed explicitly for DO-200B but using such COTS items. Like most high-quality data processing systems employed at financial institutions and factories, the use of a Quality Management System (QMS) is mandatory. DO-200B requires little additions to an already good QMS so any provably good QMS should suffice (with minor upgrades) for DO-200B. While hundreds of articles and many books exist on commercial QMS's, the following summary of QMS applies:



What is a Quality Management System?

Therefore a DO-200B compliant QMS is necessary for eventual approval of the transmitted data. Most aviation data processing entities adopt commercial-grade QMS's from ISO 9001, AS9115A, or CMMI Level 4 based systems. Attributes of a good QMS, and certainly a DO-200B

compliant QMS, are depicted in the following figure:



Aeronautical Data Assurance Level Example:

As an example, consider data associated with precision approaches. The safety assessment determined the availability of a precision approach capability would be less than a Hazardous condition but more than a Minor condition:

- a. *Catastrophic*
- b. *Hazardous*
- c. **Major**
- d. *Minor*
- e. *No Effect*

Since "Major", it is *Assurance Level 2 for DO-200B*.

Data Traceability:

DO-200B requires the defined ability to perform, and evidence to prove, thorough data traceability. Traceability for data means the ability to determine data origin. In most cases, aeronautical data users are not the originators of the data they use, manipulate, or propagate. There must be the ability to determine the source of any data error. Thus, there must be ability to trace data from any supplier to the next user and determine "root cause" of

any error. When data is corrected, subsequent users must be notified of data errors and corrections.

Timeframe, Completeness, & Format:

Data must be verified to be valid, in terms of its associated timeframe, completeness, and format. If data such as runway locations/orientation has an associated usage period of 28-days, such a period must be specified for that data and validity determined accordingly. Data completeness attributes must also be specified; for example, if data is only for use by large aircraft, then runways shorter than 3,000 feet need not be included and the data set may be complete in the absence of such short runways. Data format must be specified to determine both validity and usage; this must be in consideration of the next user on the aeronautical data chain. For example, if airport elevation data consists of two items, 1) Elevation, and 2) A sign indicating “above” or “below” sea-level, then that relationship must be defined, and the data set (elevation and above/below sea level indicator) validated accordingly.

Example of DO-200B Application:

Typically, the starting point for aeronautical data is from an Aeronautical Data Supplier who has already qualified their data using DO-201 Standards for Aeronautical Information. The DO-201 compliant aeronautical database from the aeronautical data supplier typically includes all data available from the aeronautical charts. DO-200B then is used to extract data which the particular product requires in order to perform its intended function in flight which is only a small subset of the data provided by the supplier and satisfies the “Aeronautical Data Quality Attributes” discussed above.

The data from the aeronautical database supplier is typically distributed every 28 day, for 13 cycles within a year. Much of the data contains date applicability of the data to permit changes to occur within a 28-day cycle. The DO-200B application then extracts data every 28 days from the much larger aeronautical database and distributes that data to the subscribers for data for the products they supply. DO-200B provides guidance on the qualification of the data extraction tool as well as guidance on the verification that needs to take place with each data extraction.

To meet this dual goal, most avionics system suppliers qualify their data extraction tool only when that tool changes. The data extraction tool itself accomplishes the verification steps required for every 28-day cycle extraction. Therefore, artifacts are maintained for updates to the data extraction tool separately from the data retained every time the data extraction tool is run, typically every 28 days. This data retained every 28 days from the execution of the data extraction tool must then meet the Aeronautical Data Quality Attributes Specified above.

DO-278A CNS/ATM Ground & Satellite Systems/Software

By Vance Hilderman

Reviewed by:

- Mr. Tom Roth, FAA DER
- DO-278A: Introduction.

DO-278A is often referred to as “DO-178’s little brother, for ground systems.” However, is DO-278A a little brother or more of a big sister? And is it just for ground systems? The answers may surprise ...

DO-278A is properly titled “***Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance***”. The operative term here is “CNS/ATM, which again means ground-based aviation software involved with “Communications, Navigation, Surveillance and Air Traffic Management.” While most of these systems are on the ground, some are airborne such as space-based surveillance systems and early warning and reconnaissance aircraft. For example, the latest IridiumTM communications satellite systems (which this author and 60 of his engineers worked on in the latter 1990’s) has a new generation of satellites with built-in Automatic Dependent Surveillance-Broadcast (ADSB) equipment onboard used to track aircraft; this satellite-based ADSB equipment is certified to DO-278A. (This author assisted with that effort also, though two decades later).

With such a long title, it’s a sure bet that DO-278A is NOT merely a little brother to DO-178C. In fact, DO-278 was updated to DO-278A via the Special Committee #205 and released in December of 2011. As any true student of history knows, it is not the absolute date of events which is important, but rather the context of what was occurring simultaneously elsewhere in the world which matters. In the case of SC-205, it is imperative to understand that DO-178B was being updated simultaneously which yielded DO-178C for avionics software released nearly simultaneously. The vast majority, over 95%,

of DO-278A is identical to DO-178C, for good reason: both CNS/ATM (DO-278A) and avionics (DO-178C) have potential aviation safety effects per varying criticality levels yet employ quite similar software engineering methodologies. Simply put, there was no reason to reinvent the wheel.

Since you are reading this book which is about the “Ecosystem” of aviation system development, there is no reason to repeat the DO-178C information which is found in two dedicated chapters herein but also embodied with the DO-178C applicable chapters on Tool Qualification, Model-Based Development, Object Oriented Technology, Software Quality Assurance, Software Transitions, Mistakes, and Costs. Instead, this Chapter here will highlight DO-278A fundamentals and the key differences with DO-178C.

As with airborne software (software which either executes onboard an aircraft, or directly influences the execution of such software), CNS/ATM can obviously affect aviation safety. In fact, many facets of Communication, Navigation, Surveillance, and Air Traffic Management impact safety because a single error could have dire repercussions. As a result, it is imperative that CNS/ATM be subjected to a process which has the following provable attributes:



Voilà: DO-278A is here.

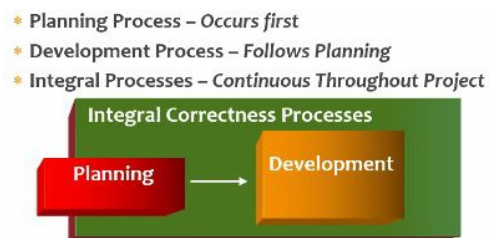
What exactly is DO-278A then? DO-278A is the second version of the baseline DO-278 document. It's a corollary to DO-178C, which as noted above is a similar standard for airborne software safety, e.g. software that typically executes onboard aircraft and which contributes to flight safety. If you already understand DO-178C, then you have the benefit of implicitly knowing 95% of DO-278A because they are similar; numerous aspects are

identical. Also, understanding that DO-278A and DO-178C are similar means you can readily tap into the much larger literature base of DO-178C, as there is relatively little published literature on DO-278A due to the relatively small number of CNS/ATM equipment manufacturers. However, there is a disadvantage in being familiar with DO-178C and wanting to understand DO-278A: human nature “glosses over” subtle differences between them which results in significant misunderstandings and mistakes when applying DO-278A. The information herein will both describe DO-278A for beginners and also illuminate differences with DO-178C.

DO-278A is a strong guideline comprising both recommendations and assessable objectives. It is intended for use in developing real-time ground or satellite based systems containing software which are involved with aircraft operations. These based systems almost always make heavy use of Commercial Off The Shelf (COTS) technologies including hardware and software. The ground-based systems governed by DO-278A often have much larger, and more diverse, software components than their airborne avionic counterparts. CNS/ATM systems have simultaneously evolved from prior systems which preceded the requirement to follow DO-278 and still employ large bases of such legacy software code. Thus the size, diversity, and increased reliance upon COTS technology all play a key role in the need for DO-278A; these CNS/ATM characteristics comprise the heart of the differences between DO-278A and DO-178C.

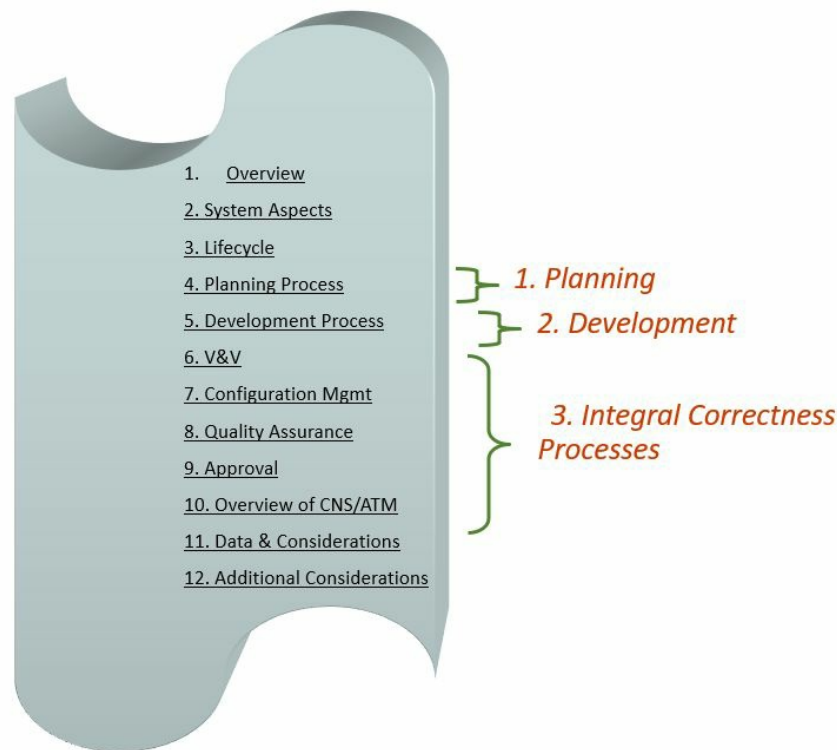
DO-278A and DO-178C: The Similarities.

Like DO-178C, DO-278A relies upon a safety assessment process to determine the system/software criticality. Where the airborne world terms this Development Assurance Level (DAL) it is shortened to Assurance Level (AL)for CNS/ATM. DO-278A has the three key processes of Planning, Development, and Integral Correctness as depicted below:



DO-278A's Three Key Processes

DO-278A's key content topics inclusive of the above Planning, Development, and Integral Correctness processes are depicted in the following diagram:



DO-278A's Key Topics versus Three Key Processes

As previously noted, DO-278A is 95% similar to DO-178C, particularly in the following areas:

- *Safety assessment with continuous involvement of Safety*
- *5 Plans, 3 Standards, and associated Checklists*
- *High and Low-level requirements*
- *Design and architecture*
- *Bi-directional Traceability*
- *Requirements & structural coverage assessment-based testing*
- *Independence of reviews at the higher criticality levels*
- *Strong configuration management including problem reporting*
- *Technical reviews and Quality Assurance audits of the above*
- *Strong Quality Assurance to assess transition criteria*

- *Tool Qualification*
- *Certification liaison*

However, there are several key differences with DO-178C. First consider the breadth of DO-278A's application domain. From its title, the primary focus upon real-time systems involved with communications, navigation, surveillance, and air traffic management is apparent. But is that all? No, and that's why DO-278A is informally referred to as the aviation standard for ground-based systems. What additional application domains might be subjected to DO-278A guidance?

- UAS ground controllers/stations (e.g. pilot stations)
- GPS equipment on the ground when in the airplane control realm
- Ground-based transceivers, including ADS-B functionality
- Satellite systems supporting aircraft communication, surveillance, and navigation

Just because an aviation-related system is ground or space-based, does that mean it must adhere to DO-278A? Not necessarily. Many ground-based systems are "involved" with aircraft or aviation but need not adhere to DO-278A. Examples of such systems include flight simulators, aircraft inventory management, flight planning, mission planning, runway lighting, and tools which assess CNS/ATM systems. The key to be considered for DO-278A application is to ask the following two questions:

1. "Does the ground-based system directly affect safety of flying aircraft without directly controlling that aircraft?"
2. "Are there outputs of the system which have not been verified by other means?"

If the answer to both of the above questions is "Yes," then the system probably falls under the DO-278A realm. Note the first question's caveat: "Without directly controlling that aircraft." What's the intention there? Consider a ground-based, Unmanned Aerial System which controls a UAV. That system has elements which directly control that aircraft by sending real-time commands; in that sense it behaves like a pilot, only the "pilot" is on the ground. But the commands to the aircraft are just as important as if the pilot

were on board the aircraft—in that case those elements may fall under DO-178C. Hence it is important to discuss the scope of the system with certification authorities early in the project's planning stage.

Consider the following statements concerning ground-based aeronautical systems and DO-278A and assess whether they are true or false; answers and explanations are provided within this chapter:

1. *T / F: All ground-based aviation systems must apply DO-278A*
2. *T / F: DO-278A is approximately 50% similar to DO-178, but many important differences exist.*
3. *T / F: AL-4 requires low-level requirements and AL-5 requires a defined architecture*
4. *T / F: AL-3 requires low-level requirements and AL-4 requires a defined architecture*
5. *T / F: Only AL-1 and AL-2 have independence requirements*

Again, if you know all the answers to the above, congratulate yourself on your above-average DO-278A knowledge as you reinforce that knowledge by affirming the answers which follow.

DO-278A's Five Plans.

As depicted in the earlier presented material on DO-278C's three key processes, the Planning process comes first, and when complete is followed by a larger Development process. In the background the largest set of Integral Correctness processes are performed continuously: Verification, Configuration Management, Quality Assurance, and Approval Liaison. Like DO-178C, a key aspect of Quality Assurance is to ensure the plans are complete and comply with DO-278A, then assessing Engineering's conformance to those plans. Also, Quality Assurance assesses the transition criteria between the requisite activities to ensure the timeliness and inputs/outputs of each activity are in accordance with plans. The five DO-278A plans are depicted below:

1.	2.	3.	4.	5.
PSAA	SQAP	SCMP	SWDP	SWVP

PSAA: Plan for Software Aspects of Approval

SQAP: Software Quality Assurance Plan

SCMP: Software Configuration Management Plan

SWDP: Software Development Plan

SWVP: Software Verification Plan

(Plus 3 DO-278A Standards: Requirements, Design and Coding)

Differences between DO-278A and DO-178C.

The first major difference with DO-278A is its emphasis upon, and allowance for, COTS systems. Whereas DO-178C places the same burden upon COTS software as for custom-development software, DO-278A recognizes that ground-based systems make extensive use of COTS technology and therefore must make accommodations for such. What would happen if DO-278A did not readily accommodate COTS software? Operating systems, graphics, database, and communications protocols are heavily used in DO-278A, extensively more than in onboard avionics via DO-178C. Furthermore, ground-based systems are much more feature-rich than airborne applications, so the software content is much greater, often 10X times larger. Since COTS technologies are generally industry neutral, they are developed without any consideration for DO-278A; thus to reverse-engineer them for DO-278A compliance would result in little value but huge cost. Instead, DO-278A is pragmatic: given the preceding, COTS technologies are allowed. However, COTS technologies within DO-278A require:

- Acquisition strategies, defined *apriori*
- Identification and analysis of verifiability
- Verification of integration and functionality
- Tight configuration management and control
- Compliance with associated DO-278A (according to Assurance Level for which COTS is used within) or justification of Alternate Means of Compliance

Note that DO-178C's certification plan was called a "Plan for Software Aspects of Certification" (PSAC). Such PSAC's are project specific, and if that project is for a specific aircraft, then that DO-178C software is both project and aircraft specific: only when it is certified for that specific project and/or aircraft is the corresponding system deemed "certified". However, CNS/ATM systems are different: Software updates and new software functionality are more continuously made to CNS/ATM systems than to aircraft. Also, CNS/ATM systems are comprised mostly of COTS hardware. Therefore, CNS/ATM systems are "approved" rather than certified and unlike aircraft hardware certification via DO-254, CNS/ATM hardware is subjected to system-level testing, not hardware-specific.

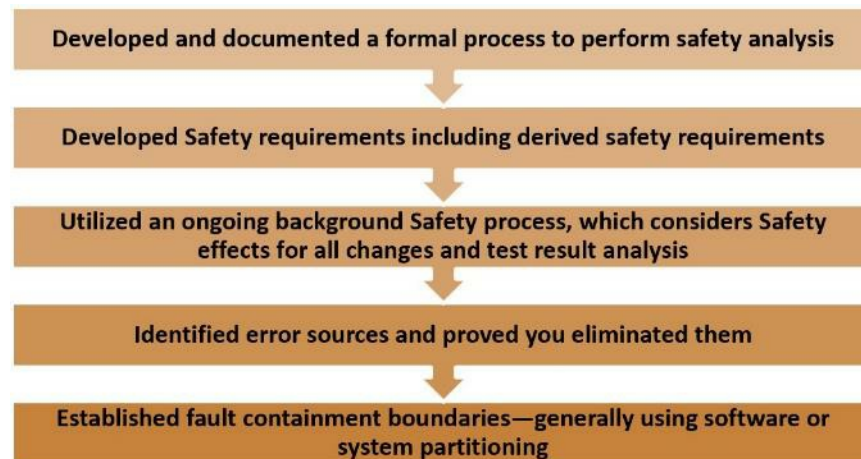
Regarding DO-278A's PSAA, this author has seen the most common PSAA mistakes denoted below:

- *Failing to justify Assurance Level and reference safety assessments*
- *Failure to describe proposed system and software architecture*
- *Failure to describe tools to be used and qualification need/strategy for tools*
- *Failure to comply with DO-278A, ED-153, Safety*
- *Failure to get certifying agency approval after submitting PSAA*

Why does DO-278A devote so much attention to COTS-related processes? Since COTS technologies are so common (and virtually mandatory) within CNS/ATM, certification authorities want to ensure a verifiable quality process is associated. The COTS quality process begins with considering which system requirements will be satisfied in whole or in part by COTS technologies. First it is necessary to actually define the ground-based system with enough granularity to perform a make-versus-buy trade-off analysis on all requirements. Those requirements allocated to "buy" become COTS designated requirements. From there, DO-278A requires consideration of the COTS lifecycle data: how will sufficient detail be provided to assess? Next, derived requirements for the COTS functionality are specified to ensure awareness of additional, or refined, capabilities which must be satisfied; those too are allocated to COTS. Then the compatibility of the COTS technology

with the target hardware/software must be considered. You must have sufficient information regarding the COTS technologies in order to address the above and proceed.

Another area of emphasis within DO-278A pertains to COTS and Safety. Since COTS is more prevalent, there needs to be an accompanying consideration of Safety vis-à-vis COTS. The manner in which faults are identified, detected, and mitigated must be addressed. Fault containment boundaries, both technical and procedural, need to be addressed. Via the safety process, error sources must be identified then efforts made to prove they were eliminated. Remember: “guilty-until-proven” innocent in aviation certification means you must keep records which prove you performed the following:



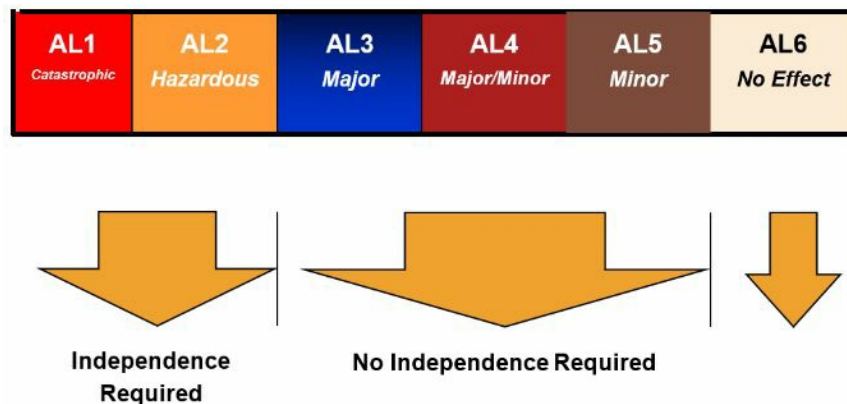
A common area of misunderstanding in DO-278A is the belief that since ground-based systems are large, complex, use COTS software, and must be cost-effective, then the costly low-level design/code processes can be skipped. **Untrue.** DO-278A at the higher assurance levels requires low-level requirements, designs, code reviews, and software structural coverage. Why? Such have been proven necessary to help assess, and thus improve, reliability and safety.

Design Assurance (Criticality) Levels:

DO-278A has specific objectives based upon the assurance level (AL) of the software. Higher AL's must satisfy more DO-278A objectives than lower levels. After the software criticality level has been determined, you examine

DO-278A to determine exactly which objectives must be satisfied for the software. Now you are ready for planning. This is where DO-278A is similar to building a house: you've performed geographic analysis to determine what type of foundation is required—that is your “safety assessment”. Then you need a Planning Process, followed by a Development Process. A concurrent Correctness Process is ongoing throughout both Planning and Development. Avionics software engineering under DO-278A is thus the same as building a house and follows the same three-phased process approach.

First, you need to understand the Assurance Level (AL) of the system and software you are developing for DO-278A. The rigor applied to planning, development, and correctness of your software is directly associated with its AL—often referred to as “criticality level.” There are six levels, with increasing rigor from Level 6 to the most stringent Level 1, as depicted below; note that “Independence” refers to the Engineering verification process, not the Quality Assurance process which is always independent:



What is meant by “Independence” above? This is “engineering” independence, not Quality Assurance independence (which is always required). Engineering independence simply means an artifact verification activity is performed by persons and processes different from those of the artifact’s author. Simple. Just remember that “Verification” equals “Reviews”, “Tests”, and “Analysis”.

Now, contrast the above with DO-178C, which is generally more well-known than DO-278A. Five of the six DO-278A assurance levels bear a close resemblance to the five assurance levels within DO-178C. The difference is

AL4, as show below:

DO-278/ED-109	DO-178/ED-12
Assurance Level	Software Level
AL1	A
AL2	B
AL3	C
AL4	No Equivalent
AL5	D
AL6	E

As can be seen in the figure above, DO-278A has a special level, AL-4, which is between DO-178's Level C and D. So, is it viewed as a level C- or a D+? The answer is both. Remember the difference between DO-178B's Level C and D? Level D had 28 objectives whereas Level C was much more rigorous with 57 objectives. That meant Level D was not concerned with how the software was designed or implemented but rather that it fulfilled an intended function while following system-level engineering processes. So DO-278A's AL-4 fits squarely in the middle: AL-4 retains a modest verification of how the software was developed. AL-4 requires data/control coupling analysis (which is design-based) but does NOT require any software structural coverage analysis or robustness testing to code as is required for AL-3.

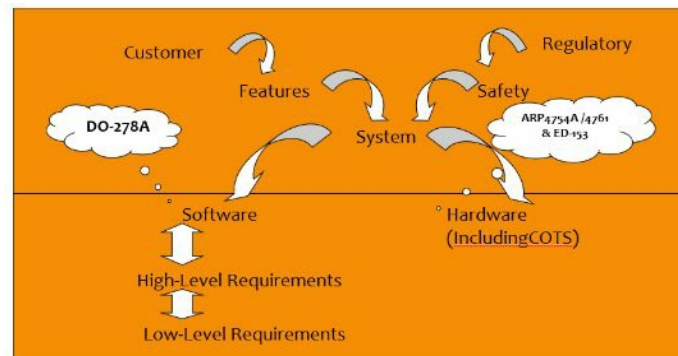
But why does DO-278 have this special in-between AL4? Think about it ... most CNS/ATM systems support safe flight but are not as directly involved with that safety as the aircraft's onboard systems. For example, if a CNS/ATM system defect should inadvertently generate instructions for a pilot to fly "through" a mountain, the aircraft's onboard systems and pilots have a high probability of detecting such erroneous instructions and disobeying them to avoid impact. Therefore most CNS/ATM software falls into the AL3 – AL5 category; very little is at AL1 – AL2. And it is this author's experience that 40-50% of CNS/ATM software is at AL4, where ample use of COTS software can be readily made because AL4 requires verification at the Application Program Interface (API) level but not the internal code and data structures level. Therefore AL4 software is vastly easier to certify than AL3. The following figure shows the different DO-278A required Objectives and independence per assurance level:

AL 1:	• 71 Objectives (30 with independence)
AL 2:	• 69 Objectives (18 with independence)
AL 3:	• 62 Objectives (5 with independence)
AL 4:	• 46 Objectives (5 with independence)
AL 5:	• 26 Objectives (2 with independence)
AL 6:	• No Objectives (just prove you are AL 6!)

DO-278A Required Objectives & Independence per Assurance Level

DO-278A's Ecosystem of Requirements

As noted in a preceding chapter for avionics software and hardware, the foundation of CNS/ATM is also based upon requirements. Whereas ARP4754A/4761A were explicitly developed for aircraft and onboard systems, their application to CNS/ATM is more adjunct. In Europe, CNS/ATM makers and EASA still have a small preference for applying ED-153, *Guidelines for ANS Software Safety Assurance*. However as of the date of this writing, ARP4754A and ARP4761 were gradually becoming more recognized for application to CNS/ATM in Europe, with non-European countries already largely moved away from ED-153. One of the challenges with ED-153 is that its myriad guidelines were deemed rather prescriptive hence subjective and more difficult to assess. However, it is also this author's opinion that anyone performing CNS/ATM safety activities read ED-153, even if not formally required, to obtain helpful information on CNS/ATM safety assessment practices.



Legacy CNS/ATM Systems, Service History, and DO-278A.

CNS/ATM Systems typically have significant non-certified legacy

software/hardware which predates more recent mandates to apply DO-278A. Instead of starting over and redeveloping these systems from scratch which could incur a heightened probability of introducing new errors, , the concept of “Service History” can be applied. Note that the “history” term is paramount: there must be provable evidence of strong record-keeping applied to that prior software/system. The Certification Authorities Software Team (CAST) Memorandum #1 covered the attributes applicable to applying service history; these are depicted below, cut/paste from CAST-1. While CAST-1 (and many other CAST papers) were removed from the FAA and EASA websites which previously contained them, CAST-1 is still applied informally to CNS/ATM systems. Note that few systems ever rank completely “Acceptable” within each of the CAST-1 Table 3.3-1 attributes, so the Plan for Software Aspects of Approval (DO-278A’s certification plan) must provide details on each attribute. The actual CAST-1 table is provided below:

TABLE 3.3-1 – Product Service History Attributes Acceptability

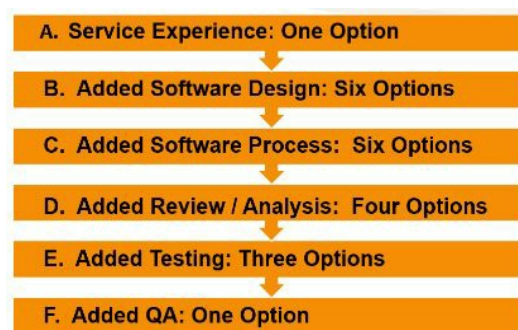
Product Service History Attribute	Not Acceptable	<-----	-----	-----	Acceptable
	--	--	--	--	>
Service Duration Length	Short	<-----	Moderate	<-----	Long
Change Control During Service	None	<-----	Marginal	<-----	Total
Proposed Use Versus Service Use	Different	<-----	Similar	<-----	Identical
Proposed Environment to Service Environment	Different	<-----	Similar	<-----	Identical
Number of Significant Mods During Service	Many	<-----	Few	<-----	None
Number of Software Mods During Service	Many	<-----	Few	<-----	None
Number of Hardware Mods During Service	Many	<-----	Few	<-----	None
Error Detection Capability	None	<-----	Some	<-----	All
Error Reporting Capability	None	<-----	Some	<-----	All
Number of In-Service Errors	Many	<-----	Some	<-----	None
Amount/Quality of Service History Data Available and Reviewed	None/ Low	<-----	Some/ OK	<-----	Much/ High
LEGEND:	No Credit Allowed	Little if any Credit Allowed	Engr. Judgment for No or Some Credit Allowed	Credit allowed based on Engr Judgment	Credit Allowed

Excerpt from CAST-1 for Product Service History

CNS/ATM systems place greater reliance upon COTS and legacy software than do their airborne counterparts. Airborne software is normally certified to DO-178C which adherents use to assess compliance for all executing software including operating systems, graphics libraries, Board Support Packages (BSP’s), etc. However, CNS/ATM system developers and certification bodies have a somewhat different philosophy: it is better to reuse proven technology, even if not developed per DO-278A, than to reinvent all new software which can then be certified to DO-278A. The rationale for this is simple: ground based systems make vastly greater use of COTS

software/hardware products; they also have vast operational code-bases which precede DO-278. As a result, DO-278A provides explicit provisions for potentially utilizing “alternate methods”; these methods then become Alternate Means of Compliance (AMC) to achieve certifiability. DO-278A’s glossary identifies a gap analysis processes to determine the gap between a proposed compliance approach and that prescribed by DO-278A. Then AMC’s are identified and analyzed for acceptability; these are then detailed within the CNS/ATM system Plan For Software Aspects of Approval (PSAA) for subsequent certification authority review and approval.

There is no formal or agreed upon AMC’s to be applied to DO-278A. However, it is this author’s experience and opinion that acceptable AMC’s are comprised from the following “toolbox” which includes the following six categories of AMC’s:



Service experience is almost universally believed to be the first, and most essential, means for alternate means of DO-278A compliance. However, service experience has a high burden of proof, and should be based upon numerous relevant criteria including service length, change control, defect density, history of usage and changes, and degree of hardware/software modifications made during that history or proposed for the new platform. These service experience criteria often exclude the viability of service experience application.

DO-331 Model-Based Development

By Vance Hilderman

Reviewed by:

- Eric Dillaber, Development Manager, Simulink Certification & Standards
- Dr Bruce Douglass, Chief Evangelist, IBM Internet of Things

DO-331 & The Four Supplements.

DO-331, *Model-Based Development and Verification Supplement to DO-178C and DO-278A*, is a 125-page guideline governing model-based development (MBD) usage in airborne and ground-based aviation software. However, since MBD is relatively new to aviation software, the authors of DO-331 faced a large hurdle: how to provide meaningful guidelines to persons generally unfamiliar with model-based development? The answer was artfully handled within DO-331 by combining workable “guidelines” with a high-level MBD tutorial which laid a common foundation for MBD terminology and application.

Ask any software developer (and their manager!) “What is the Holy Grail of software development?” and there will be one predominant answer: “software automatically generated from models with 100% reusability”. Admittedly, that answer will be provided with a smile (or a smirk, if interpreted by the engineer’s manager). But software is rarely 100% automatically generated and entirely reusable. Aviation makes ample use of software reusability and few would deny aviation software is among the best, if not the very best, in the entire world of safety-critical embedded software. Aviation authorities operate on the underlying factual premise that no software is perfect, and all software may eventually fail. That fact is unquestioned because the “answer” within aviation software is in detecting and mitigating such failures. Now, broaching modern software development practices causes one to embrace MBD as the “next great hope” in getting us a little closer to that Holy Grail: perfect and perfectly reusable software ...

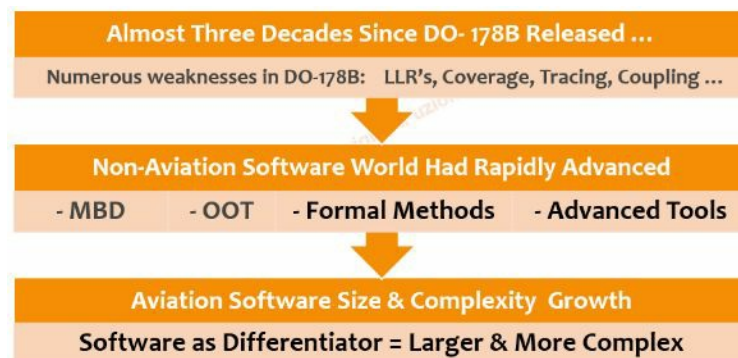
At first glance, DO-331 may seem intuitive or almost easy. Whereas DO-178C, DO-254, and DO-278A have objectives covering the entire span of aviation software and hardware engineering and generally align with other safety-critical standards, remember that DO-331 is a “supplement”. Like a nutritional supplement, DO-331 augments, not replaces, DO-178C and DO-278A. The essential Supplements for DO-XXX are:

- **DO-330:** *Tool Qualification*
- **DO-331:** *Model-Based Development (MBD)*
- **DO-332:** *Object-Oriented Technology (OOT)*
- **DO-333:** *Formal Methods (FM)*

When using any of above four technologies for aviation development, usage of the corresponding Supplement is usually mandatory. This book presents the above Supplement details in the following order since the following is the normal order of usage within aviation development:

1. **DO-331:** *Model-Based Development (MBD)*
2. **DO-332:** *Object-Oriented Technology (OOT)*
3. **DO-330:** *Tool Qualification*

Before delving further into DO-331 MBD details, first consider why these Supplements were necessary, when previously these topics had been dealt with purely informally via certification authority position papers. The figure below helps visualize how software technology evolution drove the need for the Supplements:



Why Supplements for Software Technological Evolution?

DO-331 provides a mixture of model-based development tutorial and

guidelines covering development and verification of software when using modeling. While somewhat intuitive, DO-331's real value is in the subtleties comprising provably deterministic MBD. With that, consider the following “easy” DO-331 MBD quiz:

#	Question	Answer?
1.	<i>If you use Modeling but not auto-code generation (e.g. you write code manually), do you still need a Software Modeling Standard and also Model Coverage Analysis?</i>	(Yes, No or Maybe)
2.	<i>When using models for verification, should expected results be determined prior to test execution?</i>	(Yes, No or Maybe)
3.	<i>Can the Design Model be developed from System Requirements without first decomposing System requirements to High Level Software requirements</i>	(Yes or No)
4.	<i>Must the System Requirements first be decomposed to High Level Software Requirements prior to making the Design Model?</i>	(Yes or No)
5.	<i>If System Requirements are used for the source of requirements in the Design Model, are those System Requirements then treated as High level Requirements and does the Design Model thus contain Low-Level Requirements?</i>	(Yes, No, or Maybe)

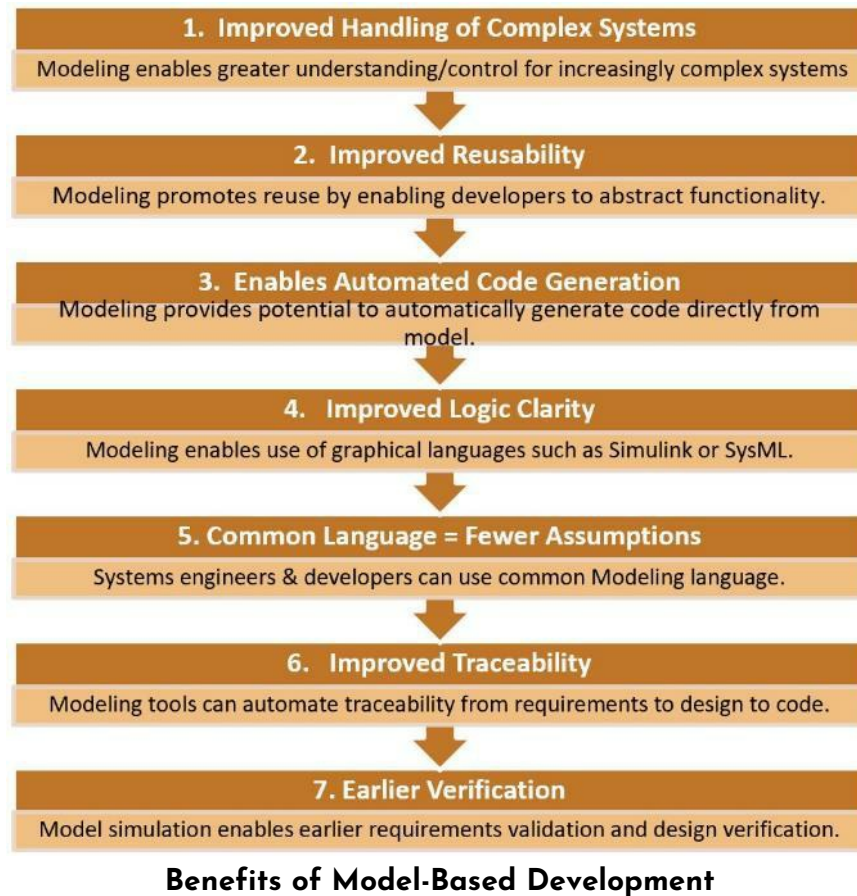
Answers to the above questions are provided within this chapter..

Why Modeling?

Software modeling is almost as old as software development, with models being actively used to assist engineers in early NASA lunar launches. Today, modeling refers to a generic activity whereby structure and behavior by and between objects is defined at a higher level abstracted from logic development. Common usage of “model” implies developing a model of structure and behavior prior to using that model to generate actual software or hardware logic. (Remember, in the avionics world, the term “hardware” includes logic embedded in silicon, previously known as “firmware” but now termed “complex electronic hardware” via DO-254). However, modeling can actually follow the logic development process instead of preceding it; many legacy codebases have modeling applied to them post-facto to aid in understanding, verifying, improving, or reusing those code bases; especially within the field of avionics whereby most systems make at least partial reuse of preceding, similar systems. This process is commonly known as “reverse

engineering.”

Why is modeling increasingly important to avionics as evidenced by the release of DO-331 in December of 2011? There are many reasons including those below:



As seen in the above Figure: “Benefits of Modeling”, there are many advantages to modeling complex systems and software. Other software domains such as telecommunications have been utilizing modeling for many years; avionics has been gradually catching up. It should be noted that the realm of avionics software certification guidance is generally devoid of cost and schedule considerations: while no one desires or expects avionics developers to go bankrupt, they are free to do so. Accordingly, what then keeps everyone from unanimously adopting modeling within their avionics development organizations? The following are often cited as reasons avionics developers avoid partial or full use of modeling:

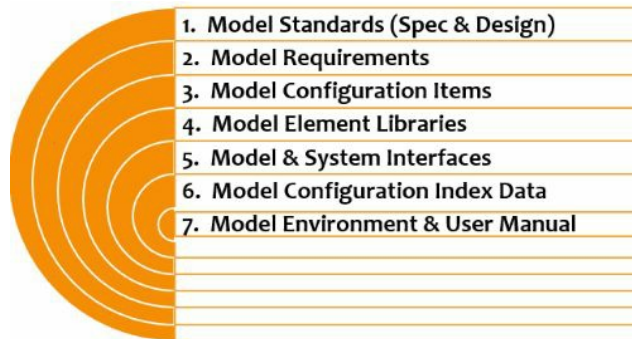
1. *Uncertainly about interpreting DO-331 or being held to an overly*

conservative interpretation;

2. *Licensing cost of modeling tools;*
3. *Learning curve of modeling tools;*
4. *Perception of difficulty or need to qualify a modeling tool per DO-330;*
5. *Culture of traditional Systems Engineers preferring to work solely with English text and avoid any modeling tool usage or input;*
6. *Fear of doing something in a new way;*
7. *A lack of trust in the quality of the source code generated from models;*
8. *A concern about the lack of control over the specific source code statements generated from models.*

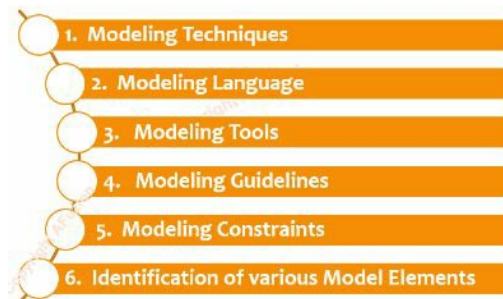
Are the aforementioned reasons for avoiding modeling valid? To the extent that perception equals reality for some people, the answer is “maybe”. But with a more informed understanding, perceptions may be changed. First, DO-331 provides a framework for enabling the usage of modeling within a prescribed regimen of modeling standards and model verification: just as for traditional software requirements and software design which must be verified to show compliance to user-defined standards, the same is true for avionics modeling. Second, modeling tool licensing costs are really not that great when considering the cost-savings obtained via engineering productivity improvements; and some modeling tools such as IBM’s Rhapsody™ and Magic Draw are quite affordable while providing the necessary key features. Third, like licensing costs, the learning curve of modeling tools is usually conquered in a matter of weeks or a few months at most; relatively short considering the multi-year duration of most avionic system developments. Fourth, it is not necessary to qualify a modeling tool: while benefits of qualification are noteworthy (such as eliminating the need for manual code peer reviews, automatic code generation, and model-level verification), all the benefits of modeling cited previously apply whether or not the modeling tool is Qualified. Fifth, all engineers are capable of learning new techniques and most actually embrace such knowledge improvements in recognition of the resume-enhancement benefits thereof. So the seeming disadvantages of modeling are readily dispelled. Furthermore, tools such as Simulink, IBM’s

Rhapsody and Ansys SCADE tools have been around for 20 years and are quite mature in both the quality of the code that is generated and in the ability to control and customize the generation of that code. However, modeling per DO-331 does require attention to key attributes summarized in the following figure and further addressed in this Chapter:



Attributes of DO-331-Compliant Model-Based Development

Of particular importance in modeling is adherence to, and subsequent assessment of such adherence to, a modeling Standard. Unlike software coding standards such as MISRA C noted in the previous chapter on DO-178C, DO-331 makes no allusion to such explicit modeling standards. However, a DO-331 compliant modeling standard must provide assessable criteria for the following key topics:



Key MBD Aspects Covered in Model Standards

Modeling Terminology

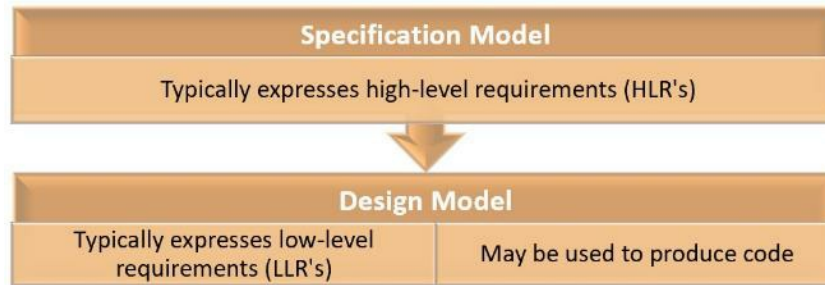
Modeling is a domain which has its own vernacular; common modeling terms are summarized below.

Code Generation	The generation of code from a design model in a specified software source code language, such as C, C++, or Ada.
Design Model	A model that defines any software design such as low-level requirements, software architecture, algorithms, component internal data structures, data flow and/or control flow. A model used to generate Source Code is a Design Model.
Model	An abstract representation of a given set of aspects of a system used for analysis, verification, simulation, code generation, or any combination thereof. It should be unambiguous, regardless of its level of abstraction. • Note: The "given set of aspects of a system" may contain all aspects of the system or only a subset.
Model-Based Development and Verification	A technology in which models represent software requirements and/or software design descriptions to support the software development and verification processes.
Model-Based Test	The creation of test cases within a modeling language and tool for the purpose of verifying the model and/or the generated source code.
Model Checking	The application of well-formedness rules to ensure syntactic correctness of the models, such as the reachability of states and the proper use of modeling language elements.
Model Coverage Analysis	An analysis that determines which requirements expressed by the Design Model were not exercised by verification based on the requirements from which the Design Model was developed. The purpose is to support the detection of unintended function in the Design Model.
Model Element	A unit from which a model is constructed.
Model Element Library	A collection of model elements used as a baseline to construct a model. A model may or may not be developed using model element libraries.
Model Simulation	The activity of exercising the behavior of a model using a model simulator.
Model Simulator	A device, computer program or system that enables the execution of a model to demonstrate its behavior in support of verification and/or validation. • The model simulator may or may not be executing code that is representative of the target code.
Modeling Technique	A combination of a modeling language and a particular manner of using this modeling language. It is driven by the level of abstraction of the information to be represented by the model and the selected modeling tools.
Report Generation	The generation of documents from a model for the particular purpose, often template-based to generate documents in a standardized format.
Reverse Engineering	The creation of a model from source code.
Specification Model	A model representing high-level requirements providing an abstract representation of software functional, performance, interface, or safety characteristics. Excludes design details such as internal data structures, internal data flow, or internal control flow.
Symbology	The graphical appearance of modeling elements. Some modeling environments allow custom symbology to be defined.
SysML	The Systems Modeling Language, a specialized variant of the UML for use in systems modeling.
UML	The Unified Modeling language, which is, by far, the most common software modeling language in use. The UML has language elements to specified software structure, behavior, functionality, and relationships.

DO-331 Modeling Terminology

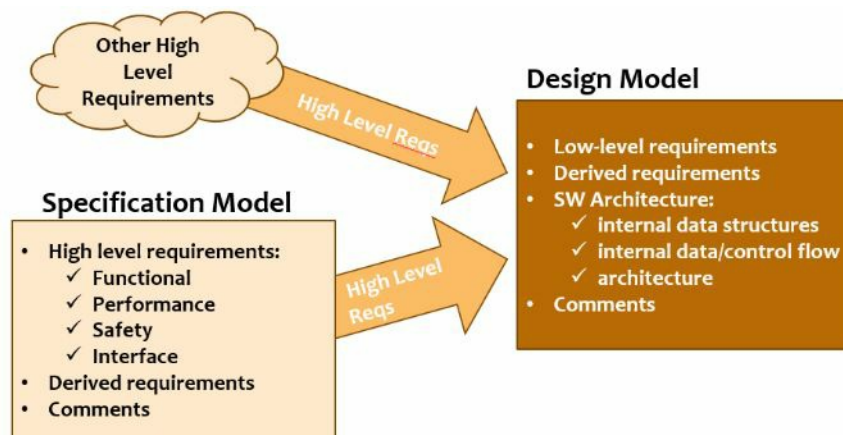
Specification Model and Design Model

It is important to recall a basic DO-178C tenet is step-wise refinement: system requirements must precede high level requirements (HLR's) and HLR's must precede low-level requirements (LLR's). The temptation to develop code directly from HLR's must be avoided. Modeling provides the ability to express HLR's and/or LLR's directly within a Model. A key facet of DO-331 modeling is the differentiation between a Specification Model and a Design Model as depicted in the following figure:



DO-331's Specification Model versus Design Model

The Design Model and the Specification Model are different, just as HLR's differ and precede LLR's: step-wise refinement. Similar to HLR's and LLR's which may reside within the same requirements specification, the Design Model and Specification Model could reside within the same model. But when both models exist, the development of the Specification Model precedes the Design Model. Since the Specification Model and Design Model accomplish different purposes, they each must use a different modeling standard.

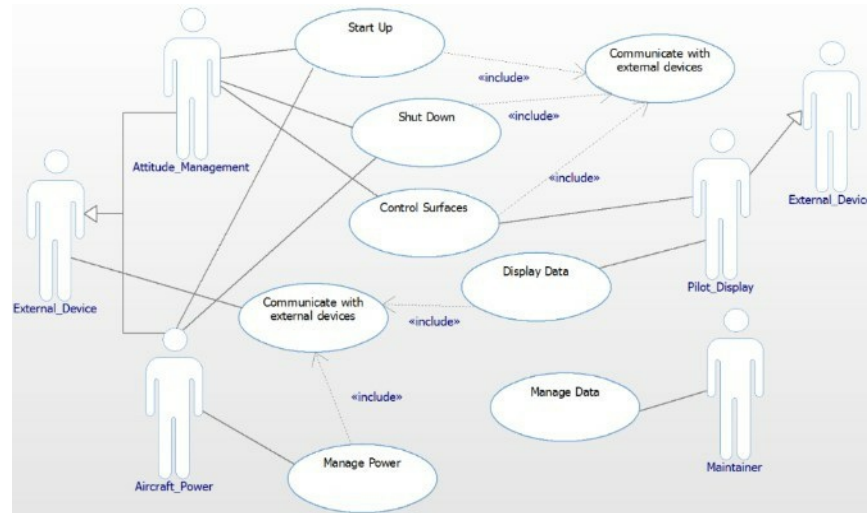


In the UML, the specification model typically employs use cases to cluster requirements, and then employs various UML mechanisms (state machines, scenario modeling, and activity modeling) to qualify and disambiguate requirements. The design model identifies the LLRs needed to meet the refined requirements and specification models. The «trace» relation supports traceability between the requirements, the specification model, and the design model.

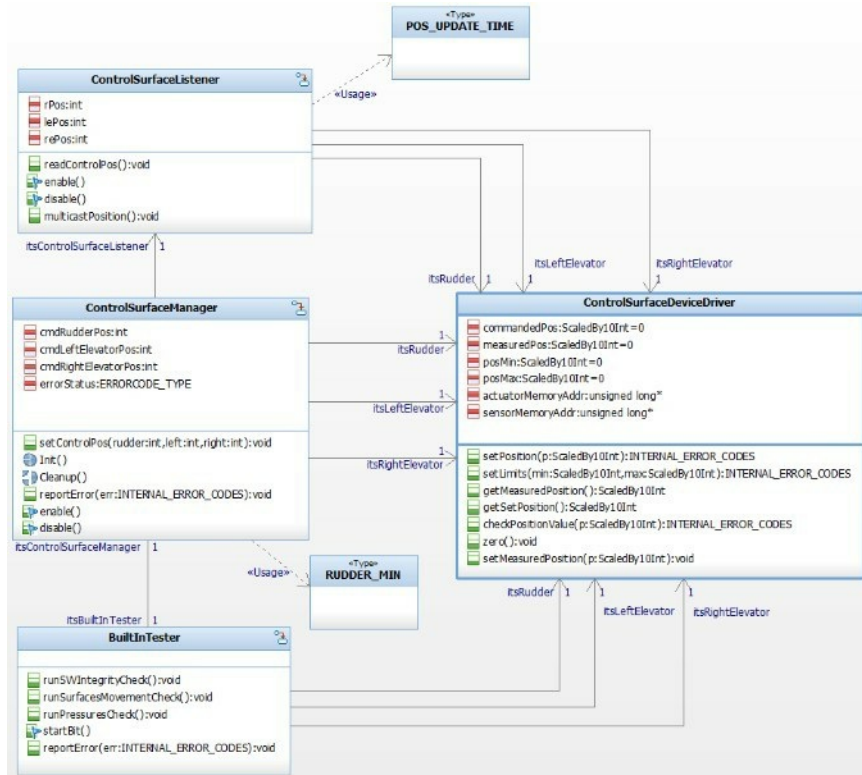
The process for model development is not specified within DO-178C or DO-331, although those standards specify the objectives with which such

processes must comply and the evidence they must produce. A common process is the Harmony Process for Embedded Software (see *Real-Time Agility* or Real-Time UML Workshop for Embedded Systems 2nd Edition, both by Bruce Powell Douglass).

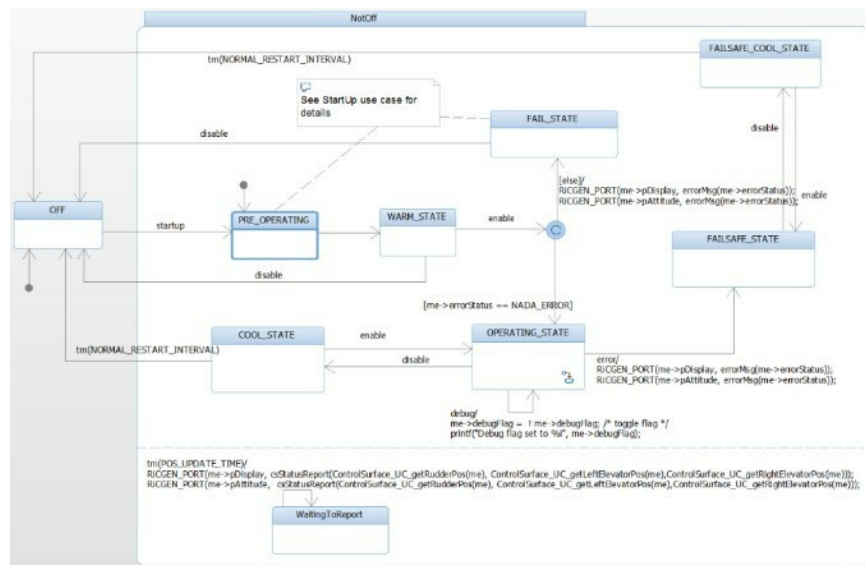
Example Use Case Diagram (reused with permission of Dr. Bruce Douglass, IBM):



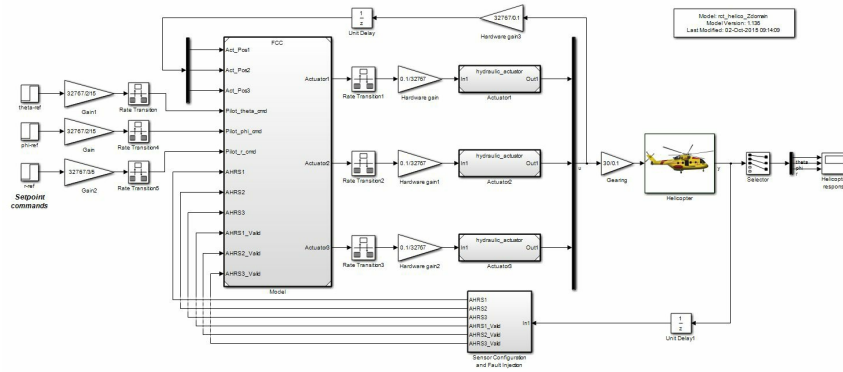
Example Class Diagram (reused with permission of Dr. Bruce Douglass, IBM):



Example State Diagram (reused with permission of Dr. Bruce Douglass, IBM):



Example Simulink Model (reused with permission of Mr Eric Dillaber, Simulink Certification Manager):



Modeling Strategy Choices

Like art, music, and gourmet dining, modeling means different things to different people and there are vastly different means to implement modeling. To narrow down the myriad modeling implementation possibilities, DO-331 describes five different modeling options and advises users to adopt one of those five. The following figure depicts the five recommended modeling options MB1 through MB5.

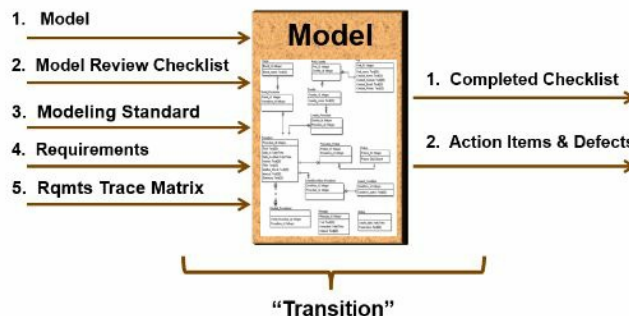
Process that generates the life-cycle	MB 1	MB 2	MB 3	MB 4	MB5
System Requirements & System Design Processes	System Requirements	System Requirements	System Requirements	System Requirements / HLR	System Requirements / HLR
Software Requirements & Software Design Processes	HLR	Specification Model (HLR)	Specification Model (HLR)	Design Model (LLR)	Design Model (LLR)
	Design Model (LLR)	Design Model (LLR)	LLR		
Software Coding Process	Source Code	Source Code	Source Code	Source Code	Source Code

As with most decisions in life, there are choices, and one size does not fit everyone. But when considering where you are from, where you are at, and where you are going, a preferred choice can be more readily ascertained. The pro/con of each modeling option is summarized below:

MB1	Traditional engineering but with a Design Model
• Pro: Introduces modeling into software development. Enables autocode generation. • Con: Separates Systems and Software engineering. HLRs independent of model.	
MB2	Traditional engineering with both Specification and Design Models
• Pro: Good use of Specification Model and Design Model. Enables autocode generation. • Con: Still has separation of Systems and Software engineering.	
MB3	Traditional engineering but with a Specification Model
• Pro: Introduces modeling for HLR's • Con: No potential for autocode generation; potential to disincentivize model upkeep.	
MB4	HLR's merged with System Rqmts; SW Engrs develop Design Model
• Pro: Promotes Systems insight and contribution to software requirements. Enables autocode generation • Con: Doesn't use a Specification Model so particularly complex systems may miss stepwise model refinement	
MB5	HLR's merged with System Rqmts; Sys Engrs develop Design Model
• Pro: Promotes strong Systems-centric development and control of HLRs and Model; very little ambiguity. • Con: Doesn't use Specification Model and possible large step between requirements and code	

Inputs to Modeling: Verifying the Model

In some software development domains, designers begin with a blank slate or concept, then iteratively evolve a corresponding software model. But in aviation, the “guilty until proven innocent” paradigm holds true: engineering work is not trusted until it is either qualified or verified. So consider: a model needs to be verified; can a model be verified merely by inspecting it? Of course not: model verification requires formal consideration of multiple inputs as depicted below:



Which of the above five inputs are actually required to perform a requisite model review? All of them, and each of the five must be under configuration management, meaning it has a unique identifier and can be retrieved in exactitude at any time in the future to determine its exact content used during the corresponding model review. Note that a common challenge in modeling is requirement verification; again, models must have requirements (software functionality) which can be used during the model review to assess the correctness and completeness of the model.

Verification through Testing

In addition to review which assesses the model's conformance to the applicable modeling standard and system/software requirements, , models themselves may be verified through testing. The UML Profile for Testing defines a standard way for test case specification, architecting, execution and analysis. This may be done by manually creating the test elements and using model simulation/execution to ensure that the outcomes match expectations. Tools such as IBM Rhapsody's add-on Test Conductor automate some of these steps and may be used as well.

Verification through Formal Methods (DO-333)

Formal methods are another key means to verify models, especially subsets of system models. Some engineers attempt to verify entire system models via formal methods, and such is ostensibly "allowed" by DO-333 (the supplement commonly applied to DO-178C and DO-278A). However, a "Best Practice" is to instead focus formal method-based model verification to a particular algorithm or set of cohesive algorithms within a model to maximize provability. (See Formal Methods for additional information).

Model Simulation

Another advantage of modeling is simulation: a model simulator may be used to execute the model earlier in the development lifecycle. Model simulation can be used to aid verification in the following ways:

- *Compliance with system requirements (Specification Models)*
- *Compliance with high-level requirements (Design Models)*
- *Accuracy and consistency*
- *Verifiability*
- *Algorithm aspects*

However, model simulation is not intended for verification of the following:

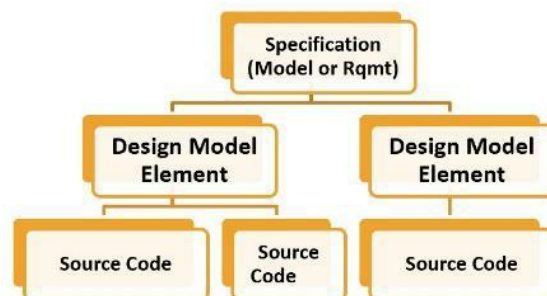
- *Compatibility with the target computer*
- *Conformance to standards*
- *Traceability*
- *Partitioning integrity*

Model and Code Verification

The UML Testing Profile (www.omg.org) provides a standard approach for specifying, executing, and analyzing test cases in the UML language. This means that all the advantages of modeling can be gained not only from the specification and design models, but also the means by which those models are verified. The IBM Rhapsody tool add-on “Test Conductor” implements that standard and comes with a qualification kit for DO-178. This tool automates test generation, execution and analysis and can provide detailed statistics about model coverage (see the next section) and, when coupled with code verification tools, code-level coverage as well.

Model Coverage Analysis & Traceability

Since the models are developed by engineers and engineers can neglect to add necessary model details or remove model details which are no longer needed, model coverage analysis must be performed. Model coverage analysis can determine which model elements may not have been completely verified and it can also detect unintended functionality, or unverified elements, within the model. Model coverage analysis may be done via simulation or formal methods; however software structural coverage is performed via actual testing. Model coverage elements are depicted below in the following diagram:



Model Elements Considered via Mandatory Model Coverage Analysis

Model traceability must also exist to prove that each model element is there for a reason. Each part of the model should be traced to:

- The requirement(s) that it implements, and
- The source code that implements it.

Model traceability can of course be performed manually, but modeling tools increasingly have built-in capabilities to support and provide traceability.

Conclusion

Modeling is a powerful capability which is increasingly applied to avionics. DO-331 provides a framework to understand modeling, harness modeling's power, and help prove the model is correct.

Answers to the questions posed earlier herein are provided below.

#	Question	Answer?
1.	<i>If you use Modeling but not auto-code generation (e.g. you write code manually), do you still need a Software Modeling Standard and also Model Code Analysis?</i>	Yes
2.	<i>When using models for verification, should expected results be determined prior to test execution?</i>	Yes
3.	<i>Can the Design Model be developed from System Requirements without first decomposing System requirements to High Level Software requirements</i>	Yes
4.	<i>Must the System Requirements first be decomposed to High Level Software Requirements prior to making the Design Model?</i>	No
5.	<i>If System Requirements are used for the source of requirements in the Design Model, are those System Requirements then treated as High level Requirements and the Design Model thus contains Low-Level Requirements?</i>	Yes

DO-332 Object-Oriented Technology (OOT)

By Vance Hilderman

Reviewed by:

- Mr. Brian Allen, Staff Software Engineer, Sikorsky - a Lockheed Martin Company

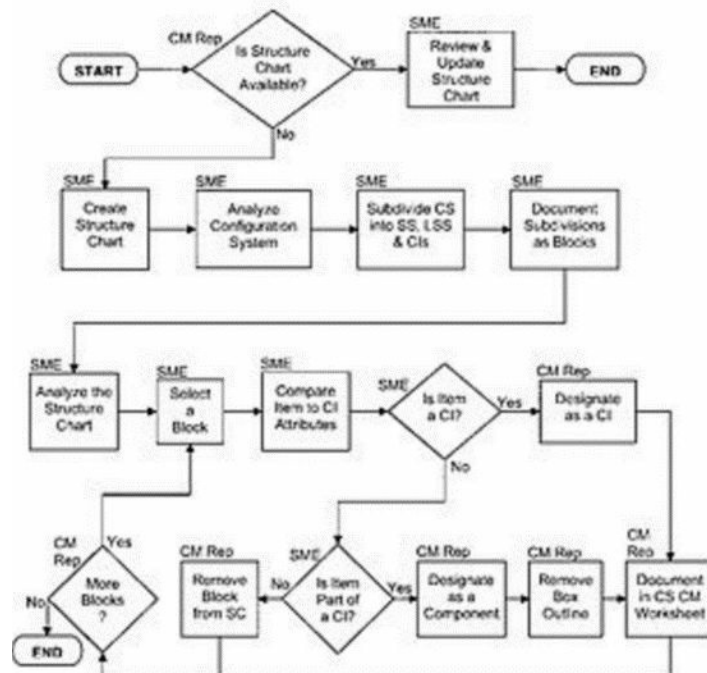
Introduction and History of DO-332/OOT

DO-332, *Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A*, is a 150-page guideline governing OOT usage in airborne and ground-based aviation software. However, since true OOT is relatively new to aviation software, (though Ada '95 has been around since ... 1995), the authors of DO-332 faced a large hurdle: how to provide meaningful guidelines to persons generally unfamiliar with Object-Oriented software? The answer was skillfully handled within DO-332 by blending practical “guidelines” with an introduction to OOT which laid a common foundation for OOT terminology, application and certifiability.

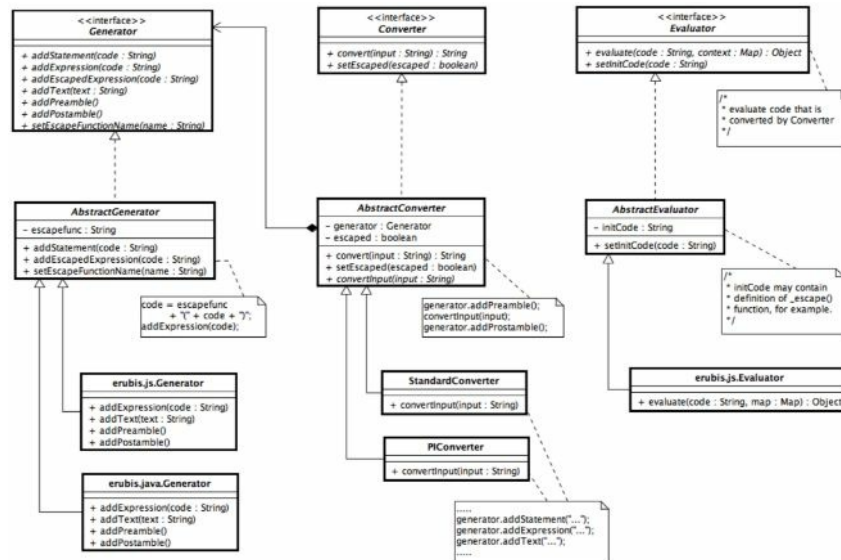
There are many ways of approaching software development; hundreds of books are in print with many seemingly preaching their own “methodology”. But as all the many colors in a peacock stem from basic Red-Green-Blue, software development at its most (overly) simplified vantage has two “primary colors”: functional structured implementation and object-oriented implementation. Unlike the peacock, with software these “colors” do not blend well; many software design elements are considered to be either “functional” or “Object-Oriented” but not both. Traditional functional software is implemented by considering then structuring each sequence of computer actions one at a time. Conversely, Object-Oriented software is designed by first articulating software objects and actions to be performed on those objects, then integrating such objects/actions into meaningful groups and events. Yes, functional design may have some objects. Yes, Object-Oriented design will use some sequential structural behaviors. But as a few drops of oil can float on water, that oil and water are hardly integrated; similarly functional and Object-

Oriented design are two distinct approaches which in their pure forms do not integrate easily with each other.

Prior to the publication of DO-332, safety-critical software developers had few rules for applying OOT. Programming standards such as MISRA C++ were available and well; those were used and should still be applied along with a commercial static analysis tool to sanitize and improve C++ source code. However, clear guidance for safety-critical OOT design and verification was lacking; DO-332 attempts to fill that void. In functional software design, the control flow is preordained by the developer thus the sequence of decisions (“control flow” in DO-178C) is considered along with the input and output data function-by-function (“data flow” in DO-178C). A typical structured sequence is depicted below:



In object-oriented software design, the individual data flow and control flow aspects are encapsulated via objects as depicted in this OOT class diagram; note the low level data/control flow elements are less visible than in the structured diagram depicted previously:



Safety-critical domains including aviation are risk-averse; new technologies are considered suspect until their safety is proven. In safety-critical software, determinism and verifiability are paramount. For many years traditionalists held that functional software development was more deterministic than OOT: structured software's execution sequencing was easily determined and repeatable. And at the unit level (software functions and collections of functions within a file), functional structured software was more readily verified: source code sections could be traced directly to associated software low-level requirements (LLR's) and tested sequence by sequence. Thus functional software design readily enabled determinism and verifiability, while doing so reliably for decades. With such a successful track record of reliability, why would anyone desire OOT with its radical paradigm shift? Simple: the very essence of *evolution* ...

According to Darwinism, evolution occurs in nature when a genetic change is seen to provide advantages for survival. Similarly for technology, evolution occurs when a change provides economic advantage, which is commercial survival. The software evolution from functional design to OOT occurred for the simple reason that OOT increasingly embodied two economic advantages over functional design: 1) greater ability to manage increasing software complexity, and 2) greater reusability. The seeds of aviation software evolution thus sprouted.

To understand the need for OOT is to understand the need for DO-332:

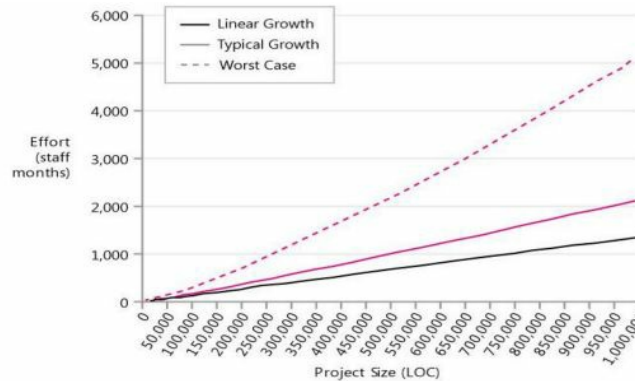
aviation software, like all software, was (and still is today) growing dramatically in size, complexity, and thus cost. Enhanced safety meant increased software functionality which meant increased software size and complexity. Functional structured software is fine, even advantageous, when functionality is simple. Computing power increased exponentially according to Moore's law allowing aviation developers to harness that increased capability.

In a recent study for an aviation client, this author analyzed the size and cost of avionics software per aircraft type, and the resulting estimates are summarized in the following table (note this data is informal and un-provable since manufacturers will not reveal financial details and much of this analysis and development was done under Non-Disclosure Agreements):

Aircraft Type	Approx. Total Lines of Software & Logic	Average Cost per Line of Software & Logic	Cost of Software Development
Piston-Powered General Aviation, 6+ seat	100K – 1.5M	\$260	\$26M - \$390M
Gulfstream G550	4.0M	\$330	\$1.32B
Boeing 777 (1995)	4M	\$260	\$1.04B
Boeing 787 (2012)	7M	\$370	\$2.59B
JSF-35 (Joint Strike Fighter, as of 2016)	15M	\$520	\$7.8B (onboard)

Avionics Software Physical & Financial Aspects per Representative Aircraft Type

The following graph (by Dr. Barry Boehm, author of the COCOMO software estimation model and a professor at USC where author received one of his Masters Degrees) depicts related software development effort based upon project size, as measured in Lines Of Code (LOC). This graph is particularly relevant because over 90% of airborne software resides in systems whose individual size varies from 50K LOC to 1M LOC, the range of this graph:



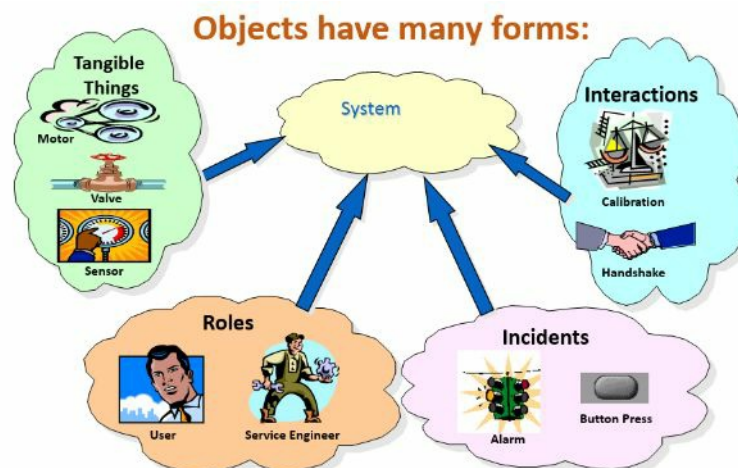
Clearly aviation software size, complexity, and cost have steadily increased along with available computing power. Also, the proliferation of different system configurations implied a greater need for software reuse. Structured software's favorable determinism and verifiability dissipates with increased system complexity while reuse is inherently challenging: seemingly small logic changes may require large analysis and rework effort to ensure continued proper operation. The commercial non-safety-critical world had recognized the power of OOT long before in implementing increasingly large and complex systems with improved reusability. Aviation enjoys leading-edge technology while eschewing bleeding-edge. In its purest form OOT embodies a variety of attributes which can obfuscate determinism and provide verification challenges. Thus OOT was regarded with suspicion. DO-178B and DO-278 preceded widespread consideration of OOT thus offered little guidance. Yet software languages such as Ada 95 embodied key OOT attributes, arguably ahead of its time. While there is little doubt OOT could improve software manageability and reusability, OOT simultaneously has features which can challenge determinism and verifiability; DO-332 is a direct response to those challenges. To better understand OOT's challenges for safety-critical system and DO-332's guidelines for compliance, one first needs a brief understanding of OOT itself.

OOT Background.

In the iron age of computing (fifty to twenty five years ago), software was written manually by conceiving the sequential instruction execution necessary to accomplish an objective. When close hardware support was needed, assembly language was favored for more direct CPU-level control; otherwise source languages such as FORTRAN, Ada, or C were typically

used for scientific programming. Aviation software grew exponentially in the 70's and 80's, meaning Ada and C predominated as the language of choice. Improving both reusability and complexity management was increasingly important, so smart developers deployed a variety of techniques toward these goals: encapsulation, hardware abstraction, wrappers, and building libraries of software components with generic and robust interfaces. While these techniques improved reusability and complexity management, the commercial consumer and financial sectors went much further: they rethought the entire premise of programming via writing sequential instructions and instead adopted Object-Oriented (OO) programming via a variety of languages designed especially for OO support. The safety-critical world slowly followed though OO posed challenges to verification, and thus certification. To understand why OO had such challenges, it's necessary to first understand OO.

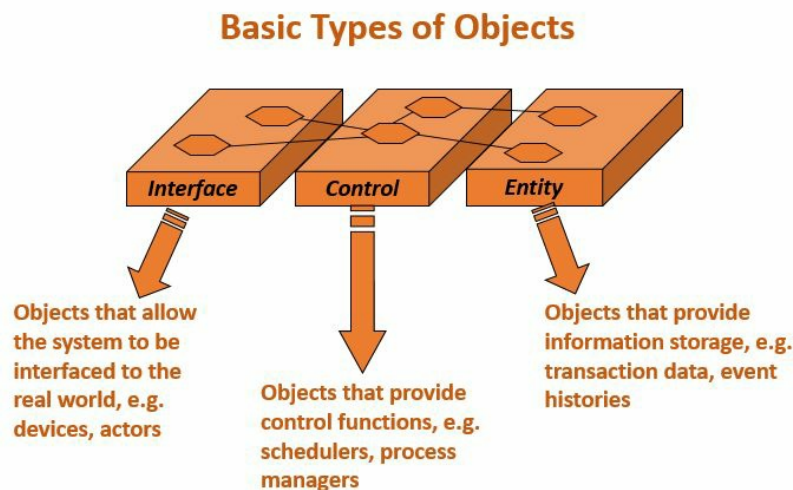
Instead of merely conceiving and writing (“coding”) sequential instructions for a computer program, OO developers think in terms of Objects. An object contains encapsulated data and procedures which are combined together and thus represent an entity. An object is a data structure that contains data. Instructions (“code”) are implemented within procedures which are called “methods”. The object has interfaces which describe how it interacts within the program. Instead of thinking in terms of individual sequential instructions, OO developers perform programming at a higher level by defining objects and interactions which consist of groups of instructions instead of single sequential instructions. An object's methods can access, and possibly update, data within the object. Objects have many forms as shown below:



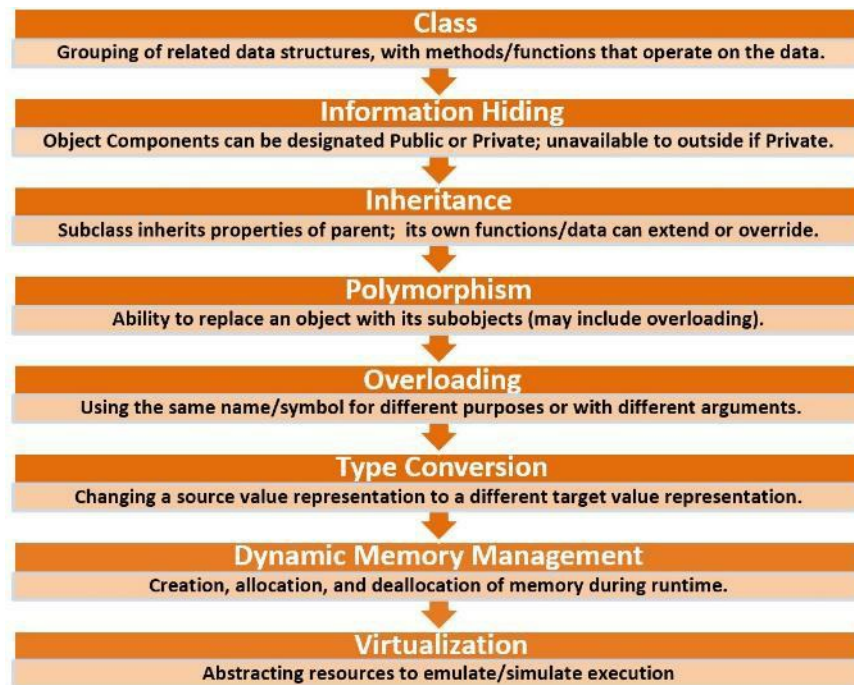
Objects then can take many different forms, but in each of these forms an object maintains the following attributes:

- Is an abstraction of a real-world concept or thing
- Has a clear boundary
- Has a unique identity
- Knows things about itself
- May perform actions on itself
- May interact with other objects

Like people, professions, and even aircraft, objects can vary dramatically in their functionality, ability, and complexity. At their most simplistic, there are three basic types of objects as depicted below:



As can be seen, objects themselves are capable and interesting. But by themselves they are not that useful. Consider an aircraft engine: by itself it too is capable and interesting. However the engine becomes powerful and useful when combined with an aircraft structure, wings, and control systems. Similarly, an object becomes powerful and useful when used within the context of Object-Oriented Programming (OOP). The basic concepts of OOP are the software design capabilities incorporated within the programming language, typically C++ for aviation and many of today's safety-critical systems. These OOT features are summarized in the figure below:



Key Object-Oriented Software Features

What is Required For OOT in Aviation?

Much of DO-332's 150-page length can be attributed to its tutorial nature; an OOT framework with definitions must be established to aid consistent compliance. DO-332 doesn't pretend to provide a full OOT tutorial though it does address the most common OOT issues pertinent to reliability and verification. Those issues largely relate to the very aspects which are unique to OOT, as summarized above. DO-332 prescribes a software engineering process which parallels that of DO-178C; namely written processes for OOT usage including OOT-related aspects added for each of the following already-required aviation software activities:

- *Plans and Standards*
- *Software Requirements*
- *Software Design*
- *Software Code*
- *Software Integration*
- *Software Verification*
- *Configuration Management*
- *Quality Assurance*
- *Certification*

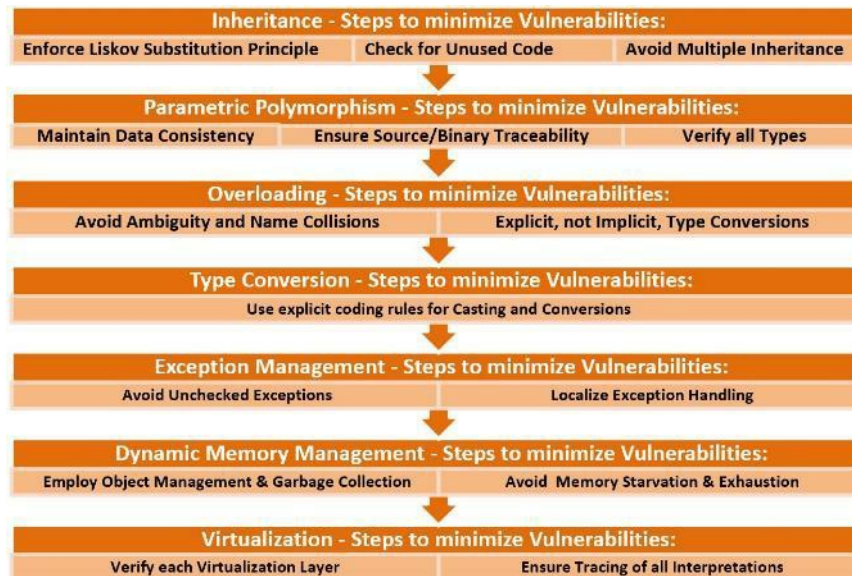
OOT within safety-critical systems must address and mitigate those aspects inherently unique to OOT which are potentially risky. Which aspects are those? Simple: those which add “*vulnerability*” to DO-178C’s requisite traceability, consistency, determinism, reviews, structural coverage, and tests.

“Vulnerability” – *Potential weakness which may reduce a system’s reliability.*

To address potential OOT vulnerabilities, DO-332 requires a regimen of assessment, prevention, and mitigation focused upon each aviation project’s specific OOT usage. Therefore a project is required to define how any OOT will be applied to airborne (DO-178C) or ground/space-based (DO-278A) runtime software. Then, vulnerability analysis specific to that project’s OOT must be performed. Projects define their OOT considerations within the following:

- Software plans (PSAC, SQAP, SCMP, SDP, and/or SVP)
- Software standards (Requirements, Design, and/or Code Standards)
- Engineering & Quality Assurance checklists

Each aviation project’s plans, standards, and checklists must comply with DO-332 therefore must reflect the results of that project’s specific vulnerability analysis. Therefore each project must consider how its OOT usage may possess vulnerabilities. Each project must consider its software development environment and lifecycle to identify potential OOT vulnerabilities; for those which could potentially exist, a mitigation then verification strategy must be defined and provably performed. DO-332 describes many common OOT vulnerabilities and their principle mitigation steps which are summarized below.



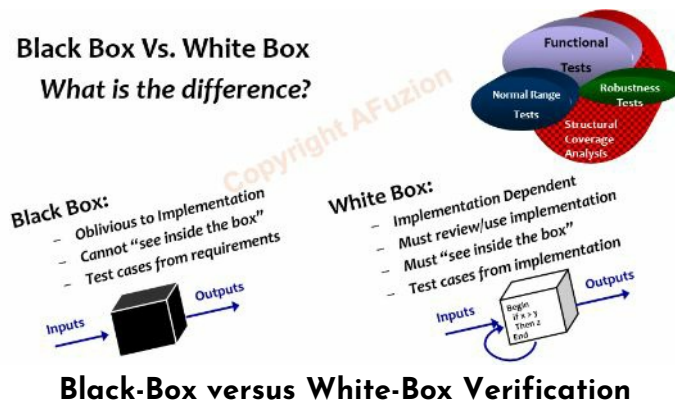
Primary OOT Aspects and Associated Vulnerability Considerations

OOT clearly has many advantages and its use is growing within safety-critical software domains. The reasons for its initial slow adoption within aviation pertain largely to the OOT vulnerabilities summarized above. Those vulnerabilities are mitigated by constraining related OOT capabilities then verifying conformance with those constraints via reviews of requirements, design, and code to associated standards. However, even if these OOT constraints could result in total elimination of vulnerabilities (and they cannot), there are still several issues associated with OOT which must be addressed; these are described below.

OOT Issues & Vulnerabilities in Safety-Critical Software

Safety-critical software must be proven to meet defined objectives. Aviation software, like most safety-critical domains, assigns an assurance level to software components based upon that software's potential contribution to a safety impact. For the higher assurance levels (DAL A-C for avionics software, DAL A-B for avionics complex electronic hardware, and AL 1 -3 for aviation CNS/ATM), those objectives require assessment of the actual logic design and implementation. This design and implementation assessment is termed "white-box" because the assessment must "see inside the box" of design and implementation. Conversely, lower assurance levels do not require white-box assessment but like the higher levels do require black-box

assessment. The figure below summarizes black-box versus white-box.



OOT's many advantages do not come risk-free. At the black-box level, there are small differences between OOT and functional software; these differences mostly pertain to software requirements which could infer a preference for an OO design including networking, communications/video utilizing messaging and packets, and dynamic or high-bandwidth asynchronous processing. However, at the white-box level, OOT's differences vis-à-vis functional software require additional consideration of the following issues, with DO-332 citing specific activities to address them.

Traceability	• Must consider inheritance including all instantiated classes and subclasses. • Must preserve traceability within models, particularly Design Model. • For DAL A, must consider source/object correlation for all OOT elements including constructors, destructors, type conversions, and compiler-generated object code.
Structural Coverage	• Via requirements-based tests, assess coverage and verify at least one instance per call point. • Show coverage of all inherited and explicit methods for each class. • Address Control Coupling and Data Coupling considerations at OOT Design and Code levels.
Component-Based Development	• Holy grail of software is "reusable components"; must ensure full verification applicable for each reuse including tracing and deactivated code. • Prove analysis of control coupling: each component in integrated system. • Verify Error Management and Resource Management in target system.
Resource Analysis	• Verify Stack and Heap usage: verify dynamic memory allocation & reclamation. • Determine Worst Case Execution Time (WCET) and verify sufficiency. • Affirm all memory operations are deterministic, with tests proving WCET. • Affirm every object reference is unique and all movements are atomic.

Key OOT Issues with Considerations to Address

DO-332 OOT Conclusion

OOT provides many advantages, especially for increasingly complex and continually evolving aviation software systems. OOT provides the ability to

greatly improve software reusability, which is the Holy Grail of software development, particularly in aviation. Essentially, when wholly using previously certified software components, only the retest and test re-reviews need be repeated; OOT greatly aids this. Perhaps the largest obstacle to software reuse is political, not technical: everyone knows that building reusable software components saves huge monies on future projects but that building such reusability also increases initial development costs. The first project pays the increased costs for designing in reusability, whereas subsequent projects reap the benefits. Smart companies would be well-advised to place more emphasis on reusability, and its longer-term cost benefits; OOT is one secret to accomplishing that. While beneficial, OOT possesses vulnerabilities which may pose risks. The analogy to human health is strong: humans can almost always benefit from improved health via cardio and strength conditioning. Amateur runners and occasional weightlifters are commonly seen in hospital emergency rooms on Saturdays where they learn that more constrained training regimens yield better and safer results. For OOT, the recommended best practices are summarized below:

- *Ensure each/every component, subclass, and interface has its own:*
 - *Requirements*
 - *Traceability*
 - *Testing (Functional, Robustness, Structural)*
- *Templates should be instantiated and tested to each type of argument, unless identical binary ensues (equivalence class)*
- *Nested templates and Friend classes prohibited for Levels A, B, & C.*
- *Ensure your Data & Control Coupling address your C++ implementation for Levels A, B & C.*
- *Read Advisory Circular (AC) 20-148 and apply its software reuse practices even when not seeking formal Certification Authority Reusable Software Component (RSC) classification.*

For software safety and health, a clear understanding of OOT with formal constraints in software development will likewise yield improved software safety and health. Start any OOT training with a thorough read of DO-332

first as it comprises an excellent tutorial in safe development.

DO-330 Tool Qualification

By Vance Hilderman

Reviewed by:

- Bunyamin Coskun, Project Manager, Milsoft Inc.
- Dave Deloria, Director of Engineering, Crane Aero.
- Roger Plieske, Rohde & Schwarz

Aviation Engineering “Tools” - Overview

Systems, Software, and Hardware engineering “tools” are computer programs that help engineers create, analyze, verify, track, modify, produce or specify the application programs being developed. Such tools and programs have been in use since nearly the beginning of computing. Yes, there are other more physical “tools” used in engineering such as oscilloscopes and items from any maintenance/installation technician’s actual toolbox; those are indeed actual tools, but they are not Tools in the sense of aviation engineering. Quite literally the “Tool” in DO-330 refers to software/logic programs which are used within the engineering development or verification of aviation systems.

Tools aid the improvement of efficiency and effectiveness in the development process by automating mundane or complex operations; they also bring the level of abstraction and understanding closer to the developer. However, can tools always be trusted or must they be formally “qualified” in some cases? The answer to that important question follows, along with details of performing such tool qualification when necessary. However, it’s not a black and white question, but rather a pragmatic exercise to determine “to what extent should we trust a tool, AND under what conditions do we need to formally qualify our tools?”

Today’s high-reliability products utilize development & verification tools within a variety of safety-critical applications; it is virtually impossible to accomplish the engineering development process without such tools. These tools may eliminate, reduce, or automate processes which ensure the

correctness of the safety-critical application. Systems for aviation, medical, railroad, space, automotive, and military industries are developed with the assistance of development tools which may contribute to faulty operation resulting in malicious behavior of the application. The development environment can affect the design and behavior of the product and must be taken into consideration. Also, a tool used to assist in verification may be incorrect, leading to an undetected error within the product.

In the aviation industry, engineering tools' potentially negative effects on avionics products must be mitigated and are regulated through the application of RTCA/DO-178C/ED-12C for software and RTCA/DO-254/ED-80 for hardware development programs respectively. And ground-based systems for CNS/ATM (Communication Navigation Systems / Air Traffic Management Systems) also have similar tool qualification needs. What do these aviation related systems have in common? They all rely upon the guidance of DO-330 for tool qualification. And in many cases per DO-330, it's not necessary to actually qualify a software tool unless the output of that tool is not otherwise verified. DO-330's tool qualification background is summarized in the following figure, and detailed in the remainder of this Chapter:



DO-330 Tool Qualification Background

What is a Tool?

Webster's dictionary defines a tool as an instrument; anything used as a means to an end, something used in the performance of an operation, anything regarded as necessary to carrying out one's occupation or profession, or a person that is used or manipulated by another (subject of another discussion, but an interesting analogy). RTCA/DO-178C, RTCA/DO-254 and FAA Order 8110.49 define a tool as a computer program used to develop, test, analyze, produce, or modify another program or its

documentation. So for avionics, a tool consists of software itself used somewhere within the lifecycle of avionics systems. Consider the following common types of tools used to develop software in the figure below, and ask yourself if they need to be qualified (to be answered in the next section):



Figure 1: Common Software Engineering Tool Types

Each of the tools in the above figure plays a key role in avionics engineering and each is available ready-to-use, as “commercial off-the-shelf” (COTS) software which can be purchased directly from any of dozens of software tool vendors. Under DO-178B, tools were simply classified as “development” tools or “verification” tools. However, DO-178C does away with such a simple classification because technical advances have allowed for hybrid tools which perform verification while also reducing subsequent development activities; this is explained later herein via tool “criteria”.

Tools used during engineering exist in all project phases: requirements specification, software design and code, integration, configuration management, and verification. Although it is possible to develop a safety-critical application in the aviation industry with the use of only implementation tools (compiler, assembler, and linker), this is increasingly unlikely given the complexity and enormity of electronic systems and modern avionics in this era.

When is It Necessary to Qualify a Tool?

Tool qualification is required whenever the design assurance process(es) described in RTCA/DO-178C or RTCA/DO-254 are eliminated, reduced, or automated by the use of the tool unless the output of the tool is verified. Verification of the tool’s output must be accomplished through the verification process as defined by RTCA/DO-178C Section 6. Remember, in

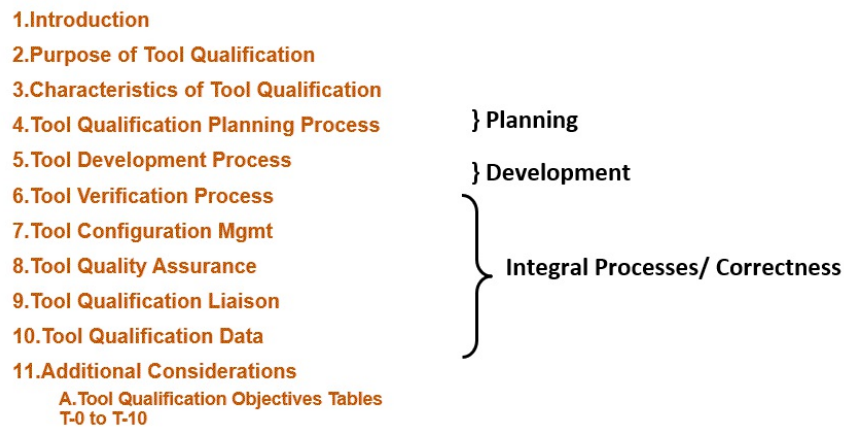
avionics development, “Verification” has a specific meaning (as the following is not official FAA/EASA policy, this author coined it and calls it the “DO-XXX Verification Equation”):

$$V = R + T + A, \text{ where}$$

$$\text{Verification} = \text{Reviews} + \text{Test} + \text{Analysis}$$

Figure 2: The DO-XXX Verification Equation

DO-330 is modeled after DO-178C in its structure (as it was released nearly simultaneously with DO-178C), and includes the similar three key Processes of Planning, Development, and Integral Correctness as shown in the Figure below:



DO-330's Organization Structure Depiction

DO-178B Versus DO-178C Tool Qualification

Under the former DO-178B (which of course is eclipsed by DO-178C), tool qualification was addressed simply within DO-178B itself and clarified via the FAA's ubiquitous 8110.49: tools were categorized as simply one of the following:

1. **Development Tools:** *capable of inserting an error within operational flight software; or*
2. **Verification Tools:** *incapable of inserting an error, but potentially capable of failing to detect an error in the flight software.*

However, DO-178B was released in 1992, in the earlier days of advanced software development, which was before associated guidelines for complex

electronic hardware (DO-254) and CNS/ATM (DO-278A) were released. The all-too-brief (less than four pages) tool qualification guidelines within DO-178B were just that: often too brief. So new avionics software tool qualification guidance was needed for multiple reasons as summarized in the following figure:



Figure 3: Reasons for DO-330 As New, Standalone Tool Qualification Guideline

As depicted above, there were four key reasons for the introduction of DO-330 “**Software Tool Qualification Considerations**” to provide the necessary supplementary tool qualification information in one document. Determining whether any tool needs to be qualified is accomplished by assessing the outcome of three questions regardless of the tool category.

1. Can the tool insert an error into the airborne software/hardware or fail to detect an existing error in the software/hardware within the scope of its intended usage?
2. Will the tool’s output not be verified or confirmed by other verification activities as specified within for example, Section 6 of DO-178C for software?
3. Will the output of the tool be used to either meet an objective or replace an objective of RTCA/DO-178C, DO-254, DO-278A, etc.?

If the answer to all three questions is YES then the tool will most likely be required to be qualified. The figure below poses these questions via a classic flow chart. It should be noted that the answer to the first and third question is almost always “Yes”, therefore the real question of tool qualification necessity normally comes down to just one question: “Will the tool output be verified?” If the answer is “No” then qualification is almost always required. A simple flow-chart for these questions is depicted below in Figure 4. To be honest, most tools can either insert an error or fail to detect an error; also most tools eliminate, reduce, or automate avionics development/verification processes. Therefore the

real question to be answered in determining if a tool needs to be qualified is “Is the output of the tool otherwise verified?” If the output of a DO-178C, DO-254, or DO-278A tool is not verified then almost always that tool must be qualified. Once you determine that a tool needs to be qualified, then you determine the tool’s TQL for your application and apply DO-330 objectives to that tool instance.

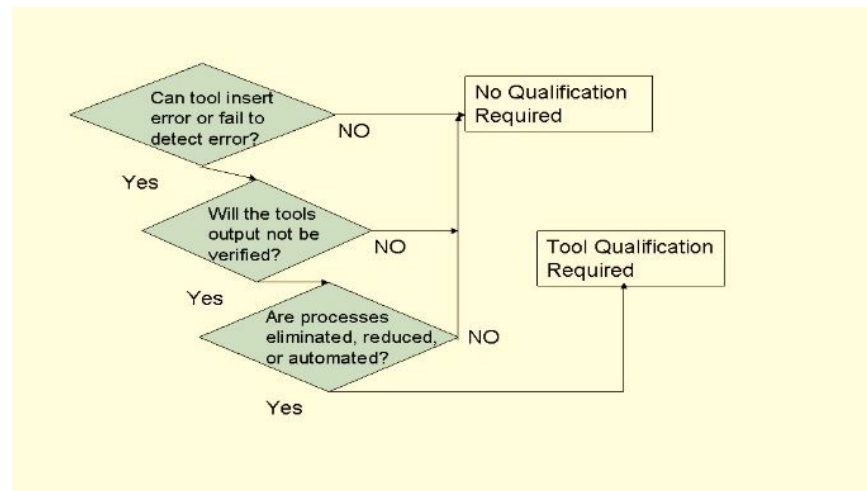


Figure 4: Simplified Flowchart to Determine Necessity of Qualifying a Tool

Why Tool Qualification?

Flight hardware, software, and systems are normally “certified”. However, tools are used in development and/or verification and the tool itself doesn’t normally fly or execute onboard the aircraft during flight. However, reliance is being placed on the tool to provide evidence and output which meet certification objectives; therefore confidence must be established to prove the tool provides at least the equivalent assurance of the certification process(es) which is/are eliminated, reduced, or automated. The dependability of the tool being used must be established. Establishing the dependability of the tool and building the confidence that the tool provides at least the equivalent design assurance process for the level required is accomplished by the tool qualification process. The first step of which is to establish whether a tool needs to be qualified as described above.

Tool qualification determination and rationale, whether or not a tool requires qualification, should be established as early as possible with the certification authority and addressed in the certification planning documents. Tool

assessment and evaluation should be performed during the planning phase of the project prior to proceeding with development and verification. If a tool is found not to require qualification, such agreement should be established early in order to avoid issues later in the project. Development may proceed prior to tool qualification being accomplished; however it's important to consider qualifiability to ensure qualification can be performed when required. This author has encountered numerous projects over the decades which “assumed” tool qualification for a given project was not required; subsequent results were disastrous when use of that tool's output was later disallowed when the tool was unable to be qualified.

DO-330 provides for tool qualification activities to be directly related to the potential tool impact; that impact is based upon both the category of the tool and also the DAL (Design Assurance Level) it's applied to. Therefore, DO-330 introduces five Tool Qualification Levels (TQLs) based upon three Tool Criteria.

Software DAL (Criticality Level)	Tool Criteria		
	1	2	3
Level A	TQL-1	TQL-4	TQL-5
Level B	TQL-2	TQL-4	TQL-5
Level C	TQL-3	TQL-5	TQL-5
Level D	TQL-4	TQL-5	TQL-5

Figure 5: Tool Qualification Levels: Based Upon DAL and Tool Criteria

There are three tool criteria, meaning a tool's usage is assessed to fall within one of the following three criteria categories:

1. **Criteria 1:** *A tool whose output is part of the airborne software and thus could insert an error.*
 - Example: code generation tool which automatically generates source code from models
2. **Criteria 2:** *A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:*

- a. *Verification process(es) other than that automated by the tool, or*
 - b. *Development process(es) that could have an impact on the airborne software.*
 - Example: model-checking tool which verifies completeness while also checking coverage
3. **Criteria 3:** *A tool that, within the scope of its intended use, could fail to detect an error.*
- Example: structural coverage tool that assesses code coverage

How is Tool Qualification Level (TQL) determined? Consider the preceding figure for Tool Qualification Level. If a tool's output, such as a code-generator, comprises DAL B software, then it's a Criteria 1 tool with a TQL of 2. On the other hand, if the tool is a structural coverage verification-only tool, used on any DAL, it is a Criteria 3 tool with a TQL of 5.

Which Tools Require Qualification?

Not all tools require qualification! By using the three determining questions previously discussed, it is relatively easy to establish which tools will or will not require qualification. Tools which typically reside in the requirements management, configuration management/data management, and quality management categories can generally be excluded from the tool qualification process. Why? Such tools generally do not supply output which either meet an objective or replace an objective of the Annex, or the tool's output is verified by another verification activity downstream of the tool's output. However, the assessment of the tool should be accomplished and will establish the need to qualify or not. The tools which generally require qualification fall cleanly into the Criteria 1, 2, or 3 categories. It is important to determine early if a tool needs to be qualified, and if so, its associated TQL. Recommendation: even if the tool assessment is thus shown to preclude qualification, cite the tool within the certification planning documents along with the rationale for why tool qualification is not required: be honest, be up-front to prevent problems downstream.

Compilers, assemblers, and linkers are typically Criteria 1 tools, but their output is often examined by another verification activity (i.e. review or testing). Therefore, they typically do not require tool qualification. Examples

of Criteria 1 tools which may require qualification include: design tools that generate source code (code generators); implementation tools that produce executable code representations; code representations or simulation tools (i.e. not actual); and binary translation tools such as cross-compilers or format generators.

Examples of Criteria 2 and 3 tools which may require qualification include: tools that automate code reviews and design reviews against standards; tools that generate test cases and/or procedures from requirements; tools that determine pass/fail status; tools that track and report structural coverage results; and tools which determine requirements coverage results.

Avionics code itself, compiler libraries, and Real-Time Operating Systems (RTOS's) are not considered tools since they form part of the actual executable software/hardware. They are verified by the design assurance process and require no tool qualification but rather full "flight software" certification per DO-178C.

Lifecycle For Qualified Tools

Quick question: can quality be built-in to a product after it's developed? Of course not: true product quality relies upon high-quality planning, implementation per plan, and assessment of implementation along with supporting processes. The basic Tool Qualification steps are shown below, in typical sequential order:

1. Determine if tool needs qualification
2. Determine which Tool Qual Criteria applies
3. Determine which Tool Qualification Level (TQL) applies
4. Identify applicable objectives & life cycle data per TQL
5. Execute tool life cycle processes
 - a. Planning process
 - b. Requirements process
 - c. Design process
 - d. Coding process
 - e. Operational integration process
 - f. Verification process
 - g. Configuration management process
 - h. Quality assurance process
 - i. Certification liaison process

Typical Tool Qualification Sequencing

Just as DO-178C requires lifecycle processes for avionics software, DO-330 defines such a lifecycle for qualified tools as shown below in the following Figure. As shown in that following figure, the tool qualification lifecycle

consists of three key activities: 1) Tool Planning, 2) Tool Development, and 3) Tool Verification; these activities must be performed sequentially, starting with Planning, then Development, and finally Verification. But continuously performed in the background during each of these activities are the corresponding Integral Processes of Tool Configuration Management, Tool Quality Assurance, and Tool Qualification Liaison.



Figure 6: DO-330 Lifecycle for Qualified Tools

Designating The TQL: Tool Qualification Level

OK, you've identified all your tools, determined which must be qualified and by what Criteria, and are ready to perform the tool qualification starting with tool qualification planning. It is critical for the planner, or planning organization, to master the organization's or project life-cycle processes, and their mapping to DO-178C/DO-254 Objectives, especially if multiple tools contribute to an Objective. But what is actually required to perform the tool qualification itself? It depends entirely on the tool's TQL. Remember, the reason for the five different TQL's is due to the simple fact that the potential adverse effect of incorrect tool usage or output varies dramatically between TQL's.: TQL 1 tool problems are normally more threatening than TQL 5 tool problems. Therefore, TQL 1 tools require the most qualification rigor and artifacts, while TQL 5 tools require the least. If a specific tool is used as a TQL 3 tool, are you allowed to qualify it to a higher level, e.g. TQL 2 or TQL1?

Yes, certainly; you're always allowed to do more work than required especially if you have unlimited budgets and schedule; but since you are intelligent, you will not do that unless you are reasonably sure you'll need to prove the higher TQL for a subsequent project. Are you allowed to qualify to a lower level, e.g. TQL 4 or TQL 5? Absolutely not, as those lower levels (ascending TQL number means lower level) have fewer qualification objectives. Recommendation: avionics development is extremely expensive and time-consuming already; avoid extra work and qualify the tool to the minimum TQL required unless you are certain you'll re-use the tool on a different project requiring that higher TQL.

Planning the Tool Qualification.

Now, the TQL has been formalized, your certification authority has formally approved your plans, or you are reasonably sure they will, so actual tool qualification can begin. The required tool qualification objectives that must be satisfied are detailed in the ten Annex A tables at the back of DO-330. These objectives depend upon the TQL, as summarized in the following figure: note that the required tool qualification objectives increase as the TQL advances from the least rigorous (TQL 5) to the most rigorous (TQL 1):



Figure 7: Key Tool Qualification Objectives & Data by TQL

The first step in planning for software tool qualification is to ensure the tool qualification is necessary, as previously explained. Presuming tool qualification is necessary, the figure above summarizes key qualification objectives and data. First, it is important to understand what objectives you

will be required to meet based upon your tool's TQL. DO-330 lists each specific objective based upon TQL but the key objectives are cited in the figure above. Again, the key point to understand is that these tool qualification objectives are additive: as TQL rigor increases, from TQL 5 to TQL 1, additional objectives are required. Is it possible that today's TQL 5 tool will be tomorrow's TQL 4 tool, or today's TQL 3 tool will be a TQL 2 tool tomorrow? Absolutely. If you even suspect that such an increase in your tool's TQL could be required, should you simply do the additional work today? Yes, if you have surplus budget and schedule; which means probably not as almost no aviation related project has such. If you think there is a reasonable probability your tool will need to be qualified to a higher TQL in the future, you should defer those additional objectives with the exception of "independence". Independence refers to the verification process (reviews, tests, analysis) and if verification was not performed with independence it would need to be done over for the higher TQL that required such. Recommendation: perform verification independently, even when not formally required; it may cost a little more for an independent engineer to gain familiarity with the technical artifacts, but the resultant independent verification will be of higher quality – just do it.

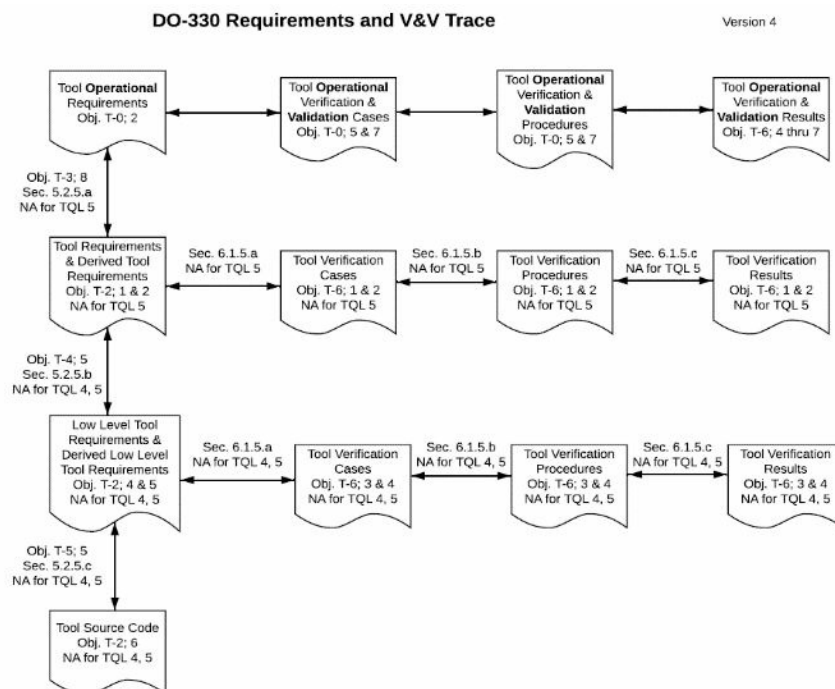
The following figure summarizes the required Tool Qualification rigor via the number of Objectives required for each TQL:

		TQL-1	TQL-2	TQL-3	TQL-4	TQL-5
T-0	Tool Operational Processes	7	7	7	7	6
	o Planning Process	1	1	1	1	1
	o Tool Operational Reqmts Process	1	1	1	1	1
	o Tool Operational Integration Process	1	1	1	1	1
	o Tool Operational V&V Process	4	4	4	4	3
T-1	Tool Planning Process	7	7	7	2	0
T-2	Tool Development Process	8	8	8	5	0
T-3	Verification of Tool Req Processes	9	9	9	8	0
T-4	Verification of Tool Design Process	11	11	9	1	0
T-5	Verification Tool Coding & Integration Process	7	7	6	0	0
T-6	Testing of Integration Process	4	4	4	2	0
T-7	Verification of Tool Testing	9	7	6	2	0
T-8	Tool CM Process	5	5	5	5	2
T-9	Tool QA Process	5	5	5	2	2
T-10	Tool Qual Liaison Process	4	4	4	4	4
Total		76	74	70	38	14

Tool Qualification Objective Synopsis per TQL

What are the Traceability Relationships for Tool Qualification?

Like the other DO-XXX guidelines, the AFuzion Incorporated training mantra of “Guilty until proven innocent: prove your innocence” reigns true for DO-330. The figure below summarizes the various provable traceability relationships which must be developed and assessed for various tool qualification requirements based upon TQL (credit Jon Lynch, AFuzion Inc DER/Engineer).



What Information & Data are Required for Tool Qualification?

The format and packaging of the tool qualification data needed to be submitted and made available for review is dependent on the type and TQL of tool being qualified. As expected, TQL 1 tools typically require the most data, whereas TQL 5 tools the least. The following Figure 8 summarizes typical tool qualification data by TQL and at what stage in the development cycle it is prepared. It should be noted that certification authorities are more concerned about the “quality” of tool qualification data than the “packaging”. Thus there is much leeway to combine tool qualification data within few documents or even within corresponding application software development

documents. While it might seem prudent to reduce the number of documents, be forewarned: good tools are leveraged on other projects or customized over time and therefore putting tool qualification data in separate documents actually simplifies re-use and re-qualification over time. And remember, the amount of data needed within each of the data items in the following figure is TQL dependent; therefore a thorough reading of all 138 pages of DO-330 should be performed to ensure the required data is included and also to avoid gathering and documenting data which is not required at less rigorous TQL's.

Why are some of the data items marked "M" (for "Maybe") in the following chart? At less-rigorous TQL's, certain data is either not required or can be placed in other documents. For example, data normally contained in a Tool Qualification Plan (TQP) for a TQL 5 tool can simply be included in the PSAC, thus negating the need for a separate TQP. Simple. One issue the planner will have to face is whether to produce one TQP per tool or produce a single consolidated TQP for all tools. Some groups opt for the later on this latest project to show how all the Tools are integrated into the life-cycle process.

Many tool vendors do not provide a Tool Qualification Certificate with a completely executed Tool Qualification Plan and a complete set of Qualification artifacts. Instead, they provide a Tool Qualification Support Package which the organization has to take ownership of, tailor, as required, then execute to produce qualification artefacts. It is critical that the organization qualifying a tool get their hands on the Tool Qualification Support Package from the tool vendor as early as possible. Typically, a Tool Qualification Support Package contains: 1) a Tool Qualification Plan template; 2) the "default" Tool Operational Requirements; 3) a Tool Verification Plan; 4) Tool Qualification Procedures and Test Cases; 5) source code files or model files required to support the execution of the Test Cases; and 6) a pro-forma Tool Accomplishment Summary. In some cases, the Tool vendor will embed tailoring instructions in the template documents, others will provide a global description document. It is important to review and assess the quality of the tailoring instructions. Organizations will also need to assess the amount of Qualification support they will require from the Tool Vendor to tailor the Tool Qualification Support Package, and sometimes, to execute it.

It should be noted the Tool Installation Report is not normally provided by the Tool Vendor and it will be critical that the organization get a good handle on this one. In some cases, it is preferable to produce a single consolidated TIR for all the tools.

DATA ITEM	STAGE			Applicable TQL				
	Liaison & Integral Processes	Tool Development	Tool Operation & Integration	TQL 1	TQL 2	TQL 3	TQL 4	TQL 5
Tool-Specific Information in PSAC	□			□	□	□	□	□
Tool Qualification Plan (TQP)	□			□	□	□	□	M
Tool Development Plan	□			□	□	□	□	M
Tool Verification Plan	□			□	□	□	□	M
Tool Configuration Management Plan	□			□	□	□	□	M
Tool Quality Assurance Plan	□			□	□	□	□	M
Tool Standards (Requirements, Design, Code)	□			□	□	□	-	-
Tool Configuration Index	□			□	□	□	□	M
Tool Lifecycle Configuration Index	□			□	□	□	□	□
Tool Problem Reports	□			□	□	□	□	□
Tool Configuration Management Records	□			□	□	□	□	□
Tool Quality Assurance Records	□			□	□	□	□	□
Tool Accomplishment Summary	□			□	□	□	□	M
Tool Info in Software Accomplishment Summary	□			□	□	□	□	□
Tool Requirements		□		□	□	□	□	□
Tool Design		□		□	□	□	M	-
Tool Source Code		□		□	□	□	□	□
Tool Executable Code		□		□	□	□	□	□
Tool Verification Cases & Results		□		□	□	□	□	□
Trace Data		□		□	□	□	□	□
Tool Operational Requirements			□	□	□	□	□	□
Tool Installation Report			□	□	□	□	□	□
Tool Operational V&V Cases and Procedures			□	□	□	□	□	□
Tool Operational V&V Results			□	□	□	□	□	□

Figure 8: Typical Custom Tool Qualification Data by TQL

= Data Item Typically Required

M = Data Item May Be Required Depending Upon Further Criteria

Blank = Data Item Typically Not Required

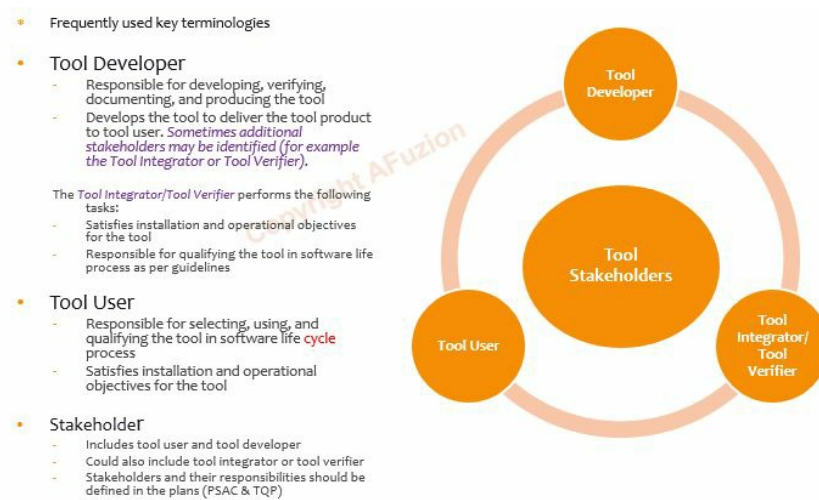
Tool Planning, Development, & Verification.

As outlined above, there are three key activities for producing qualified avionics tools:

- Tool Qualification Planning
- Tool Development
- Tool Verification

Each of these three key activities of tool qualification are described below.

But first, consider the varying roles of the Tool stakeholders: both the tool User and the tool Developer as summarized in the following figure:



Tool Stakeholders: Tool Developer vs. Tool User

1. Tool Qualification Planning & Data.

It should be no surprise that the engineering of qualified avionics tools bears strong resemblance to the engineering of avionics: “plan it”, “implement it per the plans”, and then “verify it”; all while following the Integral Processes of Planning, then Development, with CM, QA and Liaison performed in the background. Tool qualification planning has numerous objectives including:

- Determine then define the tool’s entire lifecycle and interrelationships between lifecycle processes; summarize such within a PSAC (and/or TQP).
- Identify the tool development and verification environments and related details, in advance.
- Specify applicable tool standards for requirements, design, and code; note these can be very similar to (or identical) with, similar standards for avionics software.
- Identify all applicable DO-330 objectives and define how each is to be accomplished.

The output of the tool qualification process will be applicable data including the following; :

- **Tool-Specific Information in PSAC**
- **Tool Qualification Plan (TQP)**
- **Tool Development Plan**
- **Tool Verification Plan**
- **Tool Configuration Management Plan**
- **Tool Quality Assurance Plan**
- **Tool Standards (Rqmts, Design, Code)**

2. Tool Development & Data.

AFTER the aforementioned tool qualification planning data are documented and reviewed, the tool implementation (or reverse engineering for pre-existing tools) is initiated to those plans. Applicable tool functional requirements, design, and code are developed in that order with transition and integration criteria (including traceability) affirmed and audited. Why are these important and required? Simple: to be qualified, a tool must undergo thorough verification including a minimum testing the tool's functionality versus the specified functionality in its requirements; tools with more rigorous TQL's will even have additional testing of robustness and structural coverage of the tool's source code. But testing alone can never by itself ensure high-quality. Like building a skyscraper, testing for earthquake survivability after the building is built is insufficient: such earthquake tolerances would have to be built into the building's architecture and considered throughout while building the foundation, walls, and floors. Same for avionics tools: quality provisions must be addressed throughout the development lifecycle. Hence DO-330's objectives for planning, processes, tool requirements/design/coding standards, integration, transition criteria, and traceability.

3. Tool Verification & Data.

Just as for certified aviation logic itself, the associated qualified tools used to develop or verify that certified logic require a defined verification process with specific objectives. Verification of qualified tools requires two sequential verification processes:

- a. Tool verification process, followed by the:
- b. Tool operational verification and validation process

“b” above is unique to tools: certification of avionics software (DO-178C) and hardware (DO-254) do not require operational verification and validation because the “operation” of such flight software and hardware is part of the system level requirements which are verified and validated at that system level. However, unlike avionics hardware and software, tools do not have system level requirements. Therefore, tools qualified under DO-330 must have an additional process to ensure the tool can be operated as properly intended (“verification”), and that those corresponding tool operational requirements are correct (“validation”).

Tool Requirements versus Tool Operational Requirements. Yes, qualified tools need two types of requirements: “Tool Requirements” which specify the tool’s functionality, and “Tool Operational Requirements” which specify the tool’s intended usage. Why two different types of requirements for tools? Remember, tools are commonly used by persons other than the developers of those tools. The tool developers’ work, the tool itself, needs to be verified against the intended functionality of the tool which are expressed as Tool Requirements. The tool user’s work needs to be verified against the intended usage of the tool which is expressed as Tool Operational Requirements. While there is a relationship between what a tool does versus how the tool is operated, the tool requirements are distinct from the tool operational requirements.

As a final word, it never hurts to ask for assistance when considering tools in the development process. SEEK GUIDANCE! Remember all tools must be considered, categorized, and possibly qualified. A good practice is to specify in the certification planning documents ALL tools which are to be used and whether or not qualification will be sought. Obtain agreement on tool categorization, tool qualification necessity, and tool qualification methods established early.

DO-326A Aviation Cyber-Security

By Aharon David

Reviewed by:

- Mr. Vance Hilderman, CEO/CTO, AFuzion Inc.

Aviation Cyber-Security Introduction.

The Worst Nemesis – Aviation Cyber Threats... or DO-326/ED202?

A passenger walks into a commercial-flight airplane with a laptop, hacks its network, making it fly even higher... funny? Well, unfortunately, this is not the beginning of a joke – but rather of a potential nightmare.

The bad news? This (almost as described above) allegedly happened in 2015... The good news? The person was a “white-hat-hacker” – one of the “good guys”, who only strive to prove their point about the need to strengthen cyber-defense, so the airplane was actually “safe”.

That hacker got banned from almost any future flight, but RTCA & EUROCAE didn't even require this stimulus, as by then – they were already frantically developing a solution for almost a decade. This solution, the first complete and workable DO-326/ED-202 “set” of documents was finally published in June 2018 – but even earlier than that, the FAA and EASA made the set's earlier versions as mandatory as practical at any given point in time. Following its mid-2018 publication, this DO-326/ED-202 “set” has already been made an “Acceptable Means of Compliance” (AMC), i.e.: a de-facto mandatory standard by EASA, and is currently regarded as such by the FAA, which is about to follow suit very soon.

How did we get “here” from “there”? Good question:

The “digital aircraft” is already as commonplace as the aircraft itself – as natural as aircraft wings or engines, to the extent that modern aircraft could be regarded as winged & powered computers.

The turn of the millennium saw the rise of the next aviation digital

phenomenon – the “connected aircraft”: in which everything is connected to... well... everything else... This trend is anything but novel – digital radio-systems, GPS, ACARS, ADS-B: all these, and more, have been integral components of passenger aircraft for decades. Consequently, today’s new aircraft contain thousands of processors performing both independent and intricately related operations: where old aircraft had hundreds of processors in a “closed” system, today’s aircraft architectures and connectivity necessitate more openness. However – the rapid connectivity trend, accelerated by the dramatic surge in commercial hardware and software performance, together with factors that were not previously accounted for, have made “connectivity” both a benefit-multiplier AND a menace. As a direct consequence, the DO-326/ED-202 “sets” of regulatory documents were jointly developed by the aviation industry, synchronized by RTCA (U.S.) and EUROCAE (Europe) – in order to retain the indispensable benefits of connectivity without exposing civilian aviation in general, and airworthiness in particular – to unmitigated cyber-threats that might eventually compromise safety. This “set” is already mandated for several aspects of airworthiness certification, and rapidly gaining more ground – making it an absolute necessity to get acquainted with as soon as possible for anyone in or around the business of airworthiness.

The DO-326/ED-202 “set” encompasses multiple documents containing many hundreds of pages. Therefore, a deep understanding of this ecosystem’s nature, mandates and potential trade-offs is required. Before approaching this Cyber-Security ecosystem”, it is necessary to first become acquainted with the ecosystem’s terminology and processes.

There are some major questions that need to be dealt with prior to engaging in any DO-326/ED-202 project:

- What is Cyber-Threat / Cyber-Security?
- How does it relate to Aviation/Aircraft?
- What guidance/standard for Cyber-Security exist today?
- To What Extent Are The Existing Standards Applicable to Aviation/Aircraft?
- What is “DO-326/ED-202”, and why couldn't just “ARP-4754/DO-178”

be applied?

- What is the “DO-326/ED-202 set”?
- To what extent is the “DO-326/ED-202 set” mandatory?
- What are the “326/202 set” “guidance/recommendations “(and what they are not...)?
- What does it take to meet the “326/202 set” “guidance/recommendations”?
- How can the “326/202 set” “guidance/recommendations” be efficiently met?

Honest answers to these questions are found below.

What is Cyber-Threat/Cyber-Security?

The U.S. Department of Homeland Security, in its 2010 Privacy Impact Assessment, defines “Cyber threat(s)” as “... any identified effort directed toward access to, exfiltration of, manipulation of, or impairment to the integrity, confidentiality, security, or availability of data, an application, or a federal system, without lawful authority...”

Unfortunately, everyone living in the 21st century is acquainted with this “digital menace”: computer viruses, worms, Trojan-horses and other forms of malware, have been plaguing computers since the first lab-virus. Just a quick historic perspective of this evolving digital-menace:

Phase I – “Prehistory” (pre-www):

- [1971] “Creeper”: 1st computer virus, developed by Robert H. Thomas at BBN Technologies;
- [1982] “Elk Cloner”: 1st malware to use an attack vector, originally built to combat piracy on Apple II systems;
- [1988] Morris worm: bombarded computers with traffic, could infect multiple times, brought down ~6,000 systems – Morris becomes 1st person tried & convicted under the 1986 U.S. Computer Fraud and Abuse Act;

- [1991] “Michaelangelo”, a strain of the “Stoned” virus: “wakes up” every year on March 6th (the artist's birthday), to wipe 1st 100 sectors of local hard drives & floppy disks.

Phase II – “Early History” (www-age):

- [2000] “Love Letter” worm: an email with the subject line “ILOVEYOU” and an attached “LOVE-LETTER-FOR-YOU” txt-file, infects ~50 million computers in 10 days, causing damage estimated at a few billions of USD;
- [2007] “Zeus” Trojan: a “package” containing a variety of “popular” malware programs designed for cyber thieves. The FBI arrests more than 100 hackers for bank fraud in Eastern Europe after stealing \$70M using “Zeus”.

Phase III – “Going Pro”:

- [2009] “Operation Aurora”: massive Cyber Attacks – attempts to break into- and tamper with- source code repositories from security and defense contractor companies. Affected companies include (for instance): Google, Symantec and many more;
- [2010] “Stuxnet” virus & Cyber Attacks: 1st to efficiently spy-on and tamper-with industrial-level systems.

Phase IV – “Global Menace”:

- [2013] “CryptoLocker” ransomware: cyberattack spread through email phishing scam, propagated using the Zeus Trojan. Once downloaded, ransomware payload encrypted hard drive files, “releasing” them only after ransom had been paid;
- [2016-2017] Specific attacks targeting aviation: typically aimed at airports and/or airlines, notable victims – Vietnam and Ukraine;
- [2017] “WannaCry” ransomware attack: Worldwide-synchronized attack, estimated ~\$4B damage, including infrastructure, among them Civil Aviation “actors”, such as Boeing, LATAM Airlines Group and others.

This concise history, from naïve, “experimental”, malware, to professional,

well-funded global cyber-crime today, puts at risk every aspect of modern life, from private computers to entire countries' strategic infrastructure.

So, are we all going to die? Of course we are, but probably of old age, not of cyber-crime. The reason? Cyber-Security. But what is Cyber-Security?

Cyber-Security evolved very much like cyber-threats: in a gradual manner – responsive at first, growing more professional and collaborative as threats became professional and global.

At the private, home level, the first to appear were simple protective security measures: “anti-virus”, “anti-spyware”, then – in general “anti-malware”, intended to detect and eradicate any “viruses”, “worms”, “Trojans”, and in general – any malware. A bit later, preventive measures made their debut – popularly known as “firewalls”, intended to control entry-points and block any attacks in a preemptive manner. Another type of counter-measure, although not “security-specific” was resilience and recovery, i.e.: tools that enabled at least partial/degraded operation even under attack, and tools that backed-up the system and could restore it if corrupted. These types of security measures in modern, sophisticated forms are the core of most technical security-tools even today.

However, what may be sufficient for private home-usage would hardly suffice for large organizations or facilities, so “technical security measures” became, with time, only one element among many in a more holistic approach of Cyber-Security. This approach, despite many variations among different cases, comprises the same basic principles: corporate strategy, clear roles and responsibilities, explicit codes and standards, a top-down security architecture – and eventually, yes – a variety of security measures: technical, organizational and others.

How Does Cyber-Security Relate to Aviation/Aircraft?

As long as early, amateurish, cyber-threats could be shrugged off by the Information-Technology (IT) industry as “merely annoying, but not posing any real challenge to safety”, complacency was the norm. Gradually, attackers grew bolder, malware morphed from static “beasts” to sophisticated; adjusting mechanisms with the ability to wreak havoc, and the accumulated damage grew exponentially, as previously described here.

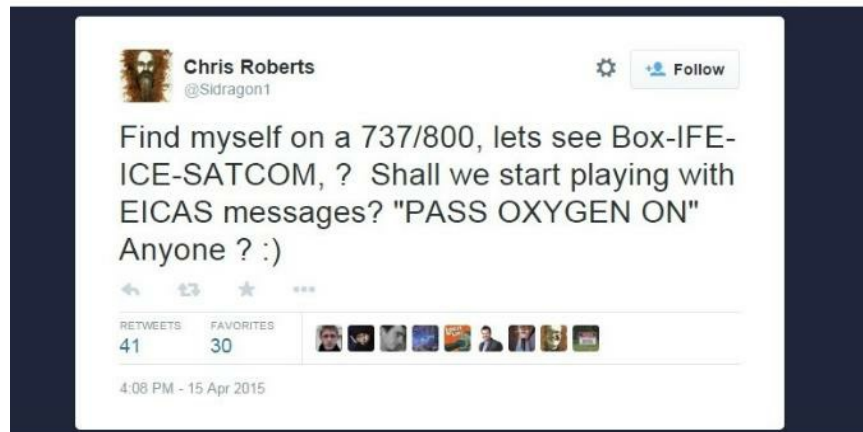
However, this trend, which sent the IT-systems community on a rapid course to devise the methods and means of establishing proper Cyber-Security, had been matched by a much slower such trend among the traditional heavy industry community, including the aviation industry, until as late as the early 21st Century. But why so?

Two main considerations led to the relative longer-lasting complacency of the traditional heavy industries towards the emerging cyber-threats: 1) the different nature of IT-systems and industrial information systems, 2) the perceived notion that such complex attacks on industrial infrastructure can only be carried out by state-level “actors”, regarded as reasonably responsible (even rogue regimes), thus, creating a “natural deterrent” for such disastrous undertakings. These two considerations were eventually proven overstated:

Industrial Information systems are indeed different from conventional IT systems; unlike IT, they arrive in a few common “flavors”, such as OT (Operational Technology), ICAS (Industrial Control & Automation Systems), SCADA (Supervisory Control and Data Acquisition), CPS (Cyber Physical Systems) and more. Typical CPS’s used to be of proprietary technology and even physically isolated from “standard” ITs (such isolation used to be referred to as “air gap”), both attributes making it virtually impossible for “mainstream” hackers to crack. However, at about the turn of the millennium, COTS hardware and software made their way slowly but surely into this solid segment, and the “connected aircraft” all but eliminated the “air gap”. As information technology and network technology exponentially evolved and their cost exponentially declined – the perceived notion of the “state-level” effort “barrier” that ensured restraint, suddenly disappeared: during the first decade of the 21st century every determined hacker could acquire (or even develop) attack tools that came “too close for comfort” for the aviation world. Even the state-decency assumption for huge attack schemes suddenly did not seem so safe, as world order destabilized and terror-organizations, criminal-organizations and even some governments drifted to the “dark side”.

As a result of the above two considerations, the second decade of the 21st century saw cyber-attacks on airports – mainly in Vietnam and Ukraine, and on civil aviation infrastructure (Boeing, LATAM and more) but the most dramatic event to demonstrate the fragility of the presumed safety was, as

implied at the beginning of this paper, this:



Fortunately, Mr. Roberts is a “white-hat hacker”, so no immediate damage – but the warning signs were loud and clear and comprised the “tipping point” catalyst for the changes leading to today’s new Cyber-Security guidelines.

What Guidance/Standard for Cyber-Security Exist Today?

Guidance and later, standards, directing organizations about the proper handling of Cyber-Security made their debut as early as the 1970s, a period when the perceived threat was still minor. After a few decades of development, literally hundreds of various types of Cyber-Security standards, guidance and best practices exist, however, three major “families” of civilian standards plus one military “family” can be pointed out as most popular:

1. Common Criteria (CC) / Common Methodology (CM), a.k.a. ISO/IEC 15408 – originally, a fusion of 3 sources: the U.S. TCSEC a.k.a. DoD 5200.28 Std. with origins in the 1970s; the Canadian CTCPEC from 1993; the European ITSEC from the 1990s and adopted in some other countries. CC/CM is a generic set of documents that mainly standardizes the terminology and methods used for Cyber-Security, providing systems/organizations assurance for their Cyber-Security claims. This set of documents does not suggest any specific (or even general) methods or solutions.
2. ISO/IEC 27000-series (a.k.a. “ISO27K”) set – this “Information Security Management System (ISMS) Family of Standards” is a

systematic, inclusive set of many dozens of specific standards, intended at covering every aspect of Cyber-Security and widely accepted as a de-facto mandatory standard around the world. Its most notable documents are 27000 – Overview and vocabulary, 27001 – Requirements, 27002 – Code of practice, 27005 – Risk Management: to mention but a few. ISO27K originates from the 1990's UK standard BS7799, based on an information security policy manual developed by the Royal Dutch/Shell Group in the late 1980s and early 1990s.

3. NIST SP-800 Series – since the 1990s, the U.S. National Institute of Standards and Technology (NIST) publishes its SP-800 series of “... information of interest to the computer security community ...” which “...comprises guidelines, recommendations, technical specifications, and annual reports of NIST’s cybersecurity activities...”, which has become an indispensable source of knowledge and served as the baseline of most other Cyber-Security standards, including the other prominent “families”. The series' documents are widely used as de-facto standards, especially for aspects that are not (yet) well covered by other standards. In addition to the SP-800 series, NIST published some major policy documents that are also highly used worldwide, such as the NIST “Cybersecurity Framework”, FIPS-200: “Minimum Security Requirements for Federal Information and Information Systems” that made the SP-800s de-facto standards, and some others.
4. DOD 8500 Series – a set of directives and instructions for the U.S. Department of Defense, aimed at “Mission Assurance”, specifically – “Information Assurance”, in which “Cyber-Security” is a tool for ensuring the performance of military missions.

All these, and many more, are used as standards and/or best-practices for various Cyber-Security purposes worldwide.

To What Extent Are The Existing Standards Applicable to Aviation/Aircraft?

While any of the major Cyber-Security standard “families” could (and do) serve as useful references for any aviation Cyber-Security standards, there were quite a few issues in which all of these standard-families came short

when aviation safety was at stake, back in 2005, when aviation Cyber-Security standardization started in earnest.

All such issues related to the nature of the major “assets” that needed to be secured by such Cyber-Security standards – the passengers on-board commercial airplanes, and accordingly, aircraft and their essential systems:

1) IT .vs. OT: Almost all aircraft systems, and many aviation control and ground-support systems, are Cyber Physical Systems (CPS) / Operational Technology (OT), as opposed to pure IT-systems. This single distinction almost precludes the usage of any major Cyber-Security standard-families previously described.

OT/CPS Cyber-Security, as previously described, had been late to get on-board, however, in 2002, the International Society for Automation (ISA) and American National Standards Institute (ANSI) launched the ANSI/ISA-99 standards committee, that started working on the “Industrial Automation and Control Systems Security” set of standards. Their baseline was the NIST SP-800-82 “Guide to Industrial Control Systems (ICS) Security”, the only existing viable OT Cyber-Security standard at that time. ANSI/ISA-99 went through a few name changes, and currently this set of standards is known as ISA/IEC-62443.

Great solution? Hardly... and for two main reasons: aviation is indeed based on OT/CPS – but also includes generic IT, and a wealth of additional considerations that cannot possibly be dealt with in a generic “one size fits all” type standard set; and, the even stronger setback – by the time the world of aviation was ready for Cyber-Security standard-setting, around 2005, ANSI/ISA-99 was not yet ready... in fact – even as late as 2020, only 8 of the 13 planned documents of the set were published, and one more part is about to be published shortly.

2) Military .vs. Civilian: Military-type standards, such as the DoD 8500-set were not adequate due to the same reason that precluded military safety standards from being adopted by civil aviation: the focus of military standards would normally be on mission accomplishment, thus – of a functional nature, while civilian standards would focus on public safety with reasonable slack for performance. Indeed – even the title of DOD 8500 is

“Mission Assurance”, so this was never really an option.

3) Other sectors: Quite a few specific Cyber-Security standards for specific sectors were developed based on generic Cyber-Security standards, and some of these sectors are even reasonably close to aviation to be seriously considered as baselines to be developed into aviation requirements, however – in 2005 no such selection existed. The closest such “relative” is SAE's J3061 “Cybersecurity Guidebook for Cyber-Physical Vehicle Systems”, published in 2016, and which is currently (2020) under a joint ISO & SAE revision process into a new standard: “ISO/SAE 21434 Road Vehicles – Cybersecurity Engineering”.

4) Existing aviation standards: In 2005, there were virtually no “existing aviation Cyber-Security standards”, at least no general-purpose and specifically – no development-phase standards. ICAO had its “Annex 17” security guide, but cyber-threats were added to it only in 2011, in two top-level paragraphs. Airlines to America (ATA), now Airlines for America (A4A) issued its “Spec 42: Aviation Industry Standards for Digital Information Security comprising about 400 pages of technical encryption guidance for aviation”, but only in 2008.

ARINC did a bit better, and in 2005 issued its “Specification 664 Part 5: Aircraft Data Network”, which was (and still is) a resourceful document for aircraft secure networking specifics. In the same year, ARINC published Technical Report 811, “Commercial Aircraft Information Security Concepts of Operation and Process Framework”, that was indeed focused on Airlines and Operations rather than development, but included the fateful recommendation:

“... Encourage the harmonization of existing aeronautical assurance practices (e.g., RTCA DO-160D and DO-178B) with security assurance practices/standards (e.g., NIST FIPS-140 and Common Criteria).

Note: The need is to bring these two domains together to provide airlines and regulators with a common ground in assessing aircraft information security solutions (e.g., evaluating COTS security solutions)...”

What Is “DO-326/ED-202”, And Why Couldn't Just “ARP-4754/DO-178” Be Applied?

As previously implied, when the European and American aviation industry via coordination with EASA and FAA, embarked on their standard-setting process circa 2005, no existing Cyber-Security standard at the time could provide a “ready-made” solution. So, in 2006 EUROCAE formed Working Group 72 (WG-72) and in 2007 RTCA formed Special Committee 216 (SC-216), both named “Aeronautical Systems Security”, and the process that yielded DO-326/ED-202 and their companion ecosystem documents, began.

The first resulting documents, ED-202 in Europe and DO-326 in the U.S., both named “Airworthiness Security Process Specification”, were published in 2010. The original DO326/ED-202 documents were intended to serve as an “all in one” guidance for the Information-Security of the development phase of aircraft from inception to certification and deployment. The original document language clearly stated that they are “...the first of a series of documents on Aeronautical Systems Security that together will address information security for the overall Aeronautical Information System Security (AISS) of airborne systems with related ground systems and environment...”, but this phase was yet to come.

DO-326/ED-202 were heavily based on the then-already-published ISO/IEC 27005 of the ISO27K family, and on the de-facto industry standard SAE ARP 4754, “Certification Considerations for Highly-Integrated or Complex Aircraft Systems”, and created a useful continuum with those in a relatively seamless manner.

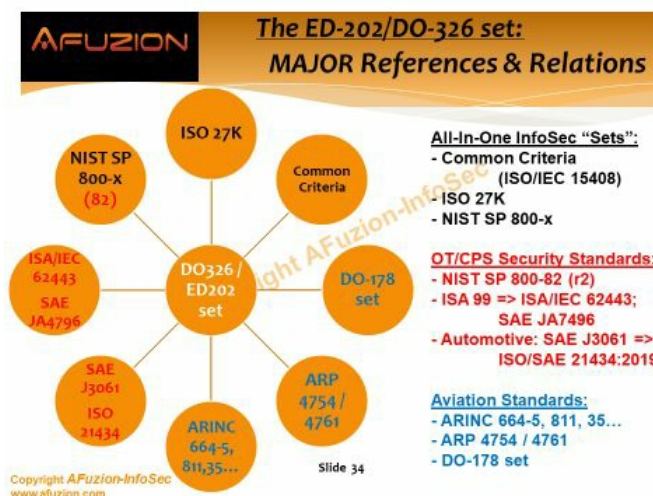
However, anyone who ever had even a brief acquaintance with ARP4754 and/or RTCA's DO-178, “Software Considerations in Airborne Systems and Equipment Certification”, should be rightly wondering why was this effort required in its entirety: why couldn't Information-Security simply made part of ARP4754 and/or DO-178, especially in light of both going through major revisions at the time?

The simple, practical, reason is stated in a DO-326/ED-202 companion document, published a few years later: “... Airworthiness security is its own discipline, needing unique expertise, and requires its own analysis techniques and assurance considerations...”, in the sense that it is distinct from safety, and from other security considerations. Another consideration was – that Information Security was not necessarily all about software – as it involves

quite a few other aspects of aviation. Therefore – ARP 4754 and DO-178 had to be coordinated with, but they also had to be kept “clean” of Cyber-Security issues.

The final deterrent from integrating Information Security into ARP 4754 and/or DO-178, was the fact that their “source of legitimacy”, FAA/EASA AC/AMC 25.1309 “System Design and Analysis”, explicitly excludes this option, by the mere definition of the term “event” covered by the document: “Event: An occurrence which has its origin distinct from the airplane, such as atmospheric conditions (e.g. gusts, temperature variations, icing and lightning strikes), runway conditions, conditions of communication, navigation, and surveillance services, bird-strike, cabin and baggage fires ... The term is not intended to cover sabotage...”. Realizing that any change to FAA/EASA formal documents may delay the process for a decade or so, and reacting to the need to close the Information-Security regulatory gap immediately – both WG-72 and SC-216 were encouraged to develop separate documents, that would be tightly coordinated with ARP 4754 and DO-178, and so they did.

And, even as the DO-326/ED-202-set is still in development, a matching “eco-system” is already being developed by SAE, at the moment – JA7496 & JA6678, which are designed to provide technical context for the DO-326/ED-202-set, as well as for automotive cyber-security standards.



What Is The “DO-326/ED-202 Set”?

As previously implied, the original DO-326/ED-202 documents were originally meant to become the first of a series for Aeronautical Information

System Security, and indeed, in the years following the 2010 initial publication, WG-72/SC-216 proceeded with the development of this “series”.

Although WG-72 were (and still are) tightly coordinated and many members share both committees, there are yet some variations between the two, mainly on the basic philosophy: whereas SC-216 (U.S.) pursues a straight forward, quick-results approach and focuses almost solely on aircraft information-security (“...document guidance for security of aircraft systems...”), WG-72 (Europe) adopted a more holistic approach, encompassing more aspects of aviation information security (“WG-72 will adopt a holistic approach, addressing security-related topics throughout the entire lifecycle of products...”). The holistic approach of WG-72 was easily recognizable right from the start, by allocating the entire ED-20x range of document numbers for the evolving series, while SC-216 simply used the sequential running-numbers of the entire DO series.

The first steps following the publication of DO-326 & ED-202 were still tightly coordinated:

1. “Clean” DO-326/ED-202 to make it a “core” document that would only cover the “WHAT” of the certification process. The revised version, DO-326A/ED-202A, was published in 2014.
2. Publish a new document, DO-355/ED-204, “Information Security Guidance for Continuing Airworthiness”, to cover the post-production, in-service phase: also in 2014. The new DO-355/ED-204 incorporated DO-326/ED-202 “spin-off” parts.
3. Publish a new companion document that addresses the “HOW” of the certification process. The new pair, DO-356/ED-203, “Airworthiness Security Methods and Considerations”, intended to be a “security DO-178” to DO-326s “security ARP4754”. Eventually, this 4754/178 equivalence was not adhered to, and both documents include parts that are parallel to both ARP4754 and DO-178.

However, at this third stage, the different nature of WG-72 and SC-216 started to take its toll, as the resulting DO-356 (published in 2014) and ED-203 (published in 2015) took different approaches, and were not technically identical like the other members of the “series”. Additionally, WG-72,

applying its holistic approach, published in 2015 a document of its own, without an SC-216 equivalent: ED-201, “Aeronautical Information System Security (AISS) Framework Guidance”, intended as a strategic, top-level document to provide a “big picture”, aided by two background reports: ER-013, “Aeronautical Information System Security Glossary” in 2015, and ER-17, “International Aeronautical Information Security Activity Mapping Summary” in 2018.

At this stage, an FAA rulemaking committee, coordinating with EASA, ANAC (Brazil) and TCCA (Canada) intervened, and among a variety of other issues approached, brought EUROCAE and RTCA committees together again, resulting in a revised DO-356 and ED-203, that were published in 2018 as DO-356A/ED-203A, unifying their technical content. However, ED-201 was not matched by a DO equivalent, as it was not deemed a “working document” but an “orientation document”.

Meanwhile, WG-72 went on to publish, in 2019, another document of its own, ED-205, “Process Standard for Security Certification/Declaration of Air Traffic Management/Air Navigation Services (ATM/ANS) Ground Systems”, again – without an upfront SC-216 equivalent, however, as of December 2020, RTCA decided to join this effort with its own version.

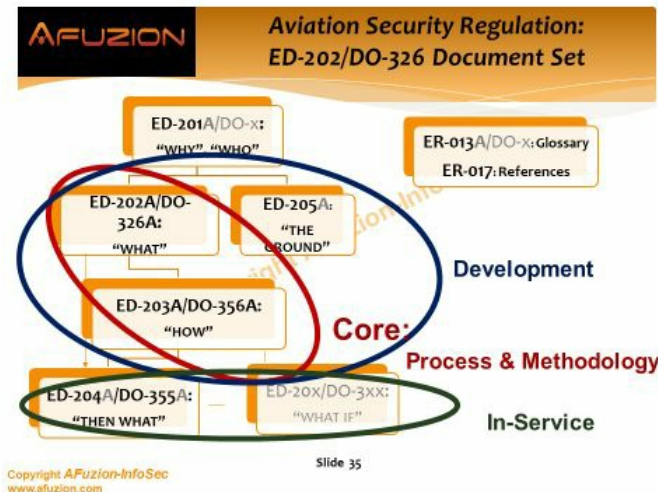
Thus, as of 2021, the DO-326/ED-202 set comprises:

1. The “core” guidance and considerations, DO-326A/ED-202A and DO-356A/ED-203A;
2. The “in-service” guidance, DO-355/ED-204, based on the “core” guidance;
3. The “top-level” documents, ED-201, ER-013, ER-017, serving as “philosophy guidelines”;
4. The ground systems standard, ED-205, mainly intended for European usage.

During the next couple of years, WG-72 / SC-210 plan to produce:

1. A new DO/ED “Information Security Event Management” companion document to support DO-355/ED-204;

2. Revision A for DO-355/ED-204, which has already published, but still needs to be included in the formal regulation in Europe and the US;
3. Revision A for ED-201, and a new DO, equivalent to ED-201A;
4. 2) Revision A for ED-205, and a new DO, equivalent to ED-205A.
5. Revision A for ER-013, and a new DO, equivalent to ER-13A.



To What Extent Is the "DO-326/ED-202 Set" Mandatory?

In Europe, on July 1st, 2020, EASA made public its "ED Decision 2020/006/R", adopting, with minor revisions, EASA's February 2019 Notice of Proposed Amendment (NPA) 2019-01, "Aircraft cybersecurity, **effective Jan 1st, 2021**."

ED Decision 2020/006/R amends CS-25, CS-27, CS-29, CS-APU, CS-E, CS-ETSO, CS-P, the related Acceptable Means of Compliance (AMC) and guidance material (GM), together with AMC-20, AMC/GM to CS-23 and AMC/GM to Part 21.

The amendment to AMC-20, adding AMC 20-42 "Airworthiness information security risk assessment", practically makes the core documents of the ED-202-set (and their equivalent, the U.S. DO-326-set) – the **ONLY** means of compliance for the cybersecurity aspects of airworthiness certification:

Another EASA "Rule Making Task", RMT.0720 (Aviation System Cybersecurity), already issued proposed amendments to existing regulation, NPA 2019-07, for the entire aviation sector, to complement the airworthiness

regulation mentioned above, which is expected to be accepted with some revisions in the 2021-22 time frame and make ED-201 and ED-205 “Acceptable Means of Compliance” as well.

In the U.S., the previously mentioned FAA rulemaking committee, FAA Aviation Rulemaking Advisory Committee (ARAC) Aircraft System Information Security / Protection (ASISP) Working Group (WG), that during 2015-2016 examined the entire information security regulation, considerably accelerated the process of establishing formal, mandatory, regulation, and the very clear recommendation on this was for the FAA to “...consider RTCA standards DO-326, DO-356 and DO-355 and EUROCAE standards ED-201, ED-202, ED-203, ED-204 as acceptable guidance materials to comply with the security rule 25.13xx for large transport aircraft for new Type type-certifications or new significant major changes or when the applicant elects to use them on a voluntary basis...”, which in regulatory parlance translated to “FAA: make it mandatory, now”. As EASA (Europe), ANAC (Brazil) and TCCA (Canada) actively cooperated with the FAA ARAC ASISP WG, the clear meaning of this is – the DO-326/ED-202 set is becoming mandatory, and very soon, all over the world. This recommendation is well harmonized with EASA's task force RMT.0648, developing the “Certification Specifications for product design” and task force RMT.0720, developing the “Rules for risk management within organizations and service providers” – both “horizontal” guidance, designating the DO-326/ED-202 set, scheduled to be published in 2020.

Other critical ARAC ASISP WG recommendations were to retroactively carry out a “Table-top Review” of **existing** Airborne CNS/ATM TSOs and standards and establish guidance for the Use of Commercial Off The Shelf (COTS) and **Previously Certified** Products.

As the ARAC ASISP WG also recommended that the DO-326/ED-202 set should be adapted to all categories of aircraft, engines and propellers, together with very explicit language for new rules and regulations, the very clear answer is: everyone in the aviation business should be ready for the DO-326/ED-202 to be completely mandatory within no more than a decade, possibly – much sooner.

Some aspects of the DO-326/ED-202 set that are already mandatory:

- FAA Policy Statement PS-AIR-21.16-02 Rev. 2, “Establishment of Special Conditions for Aircraft Systems Information Security Protection” from 2017, states that the FAA “...will issue special conditions for initial type certificate (TC), supplemental type certificate (STC), amended TC, or amended STC applications for aircraft systems connecting to non-trusted services (e.g., non-governmental) and networks...”, for all aircraft categories: part 25 Transport, part 23 Commuter Category, part 27 Multi Engine Normal Category Rotorcraft and part 29 Transport Category Rotorcraft. As the said revision 2 is the product of the ARAC ASISP WG, and as the DO-326/ED-202 set was recommended as AMC, the meaning of this policy is that de-facto, the DO-326/ED-202 set is now the norm for this type of certification, for all aircraft categories.
- FAA Advisory Circular AC 119-1, “Airworthiness & Operational Authorization of Aircraft Network Security Program (ANSP)”, from 2015, uses the “in-service” part of the set: DO-355, which relies on DO-326 & DO-356.
- FAA Advisory Circular AC 20-140C, “Guidelines for Design Approval of Aircraft Data Link Communication Systems Supporting Air Traffic Services (ATS)”, from 2015, uses the “core” and the “in-service” full set: DO-326, DO-356 & DO-355.
- Advisory Circular (AC) 120-76D: “Authorization for Use of Electronic Flight Bags”, from 2017, includes a new “Security Procedures” section which uses (AC) 20-140C, which relies on the entire DO-326 set.

Note that more mandates are already here, so ignore the DO-326/ED-202 set at your peril...

What Are The “DO-326/ED-202 Set” “Guidance/Recommendations”? (...and What They Are Not...)

The DO-326/ED-202 set's guidance and recommendations are dispersed across all the documents comprising the set, but for airworthiness purposes, it can be regarded as a 2-part set: (1) DO-326/ED-202 & DO-356/DO-203, the “core” addressing the development phase, (2) DO-355/ED-204, the “continued airworthiness” addressing the “in-service” phase. In order to

properly comply with the set – it would be best regarded by applicants as ... a set – one unified body of text, for development, service or both – rather than an eclectic collection of documents...

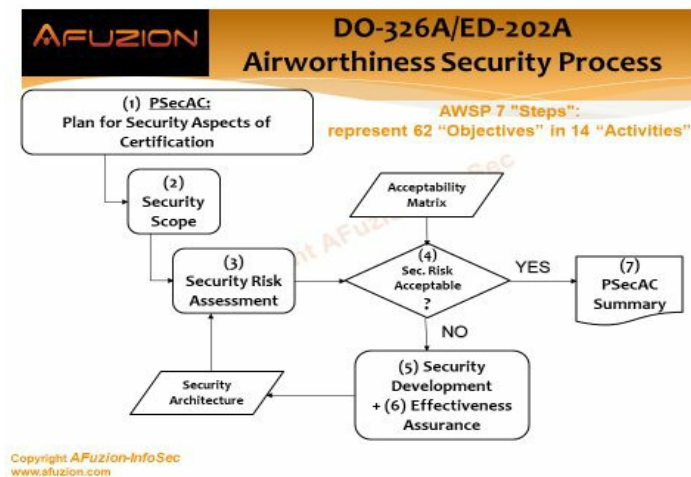
Neither part nor any specific document of the set dictates any specific security measures, techniques or methods to be deployed – so by no means should the DO-326/ED-202 set regarded as a “cook book”, but rather, as their titles imply, guidance, ranging from top-level strategy to detailed tactics, which would, in many cases, necessitate applying further information security standards: some are even explicitly recommended by the set, e.g. ISO 27K.

As for the guidance/recommendations that are included in the set, these can be roughly described as:

1) The “Airworthiness Security Process” (AWSP), mostly detailed in DO-326A/ED-202A, that outlines the major steps, activities and objectives of security certification for airworthiness:

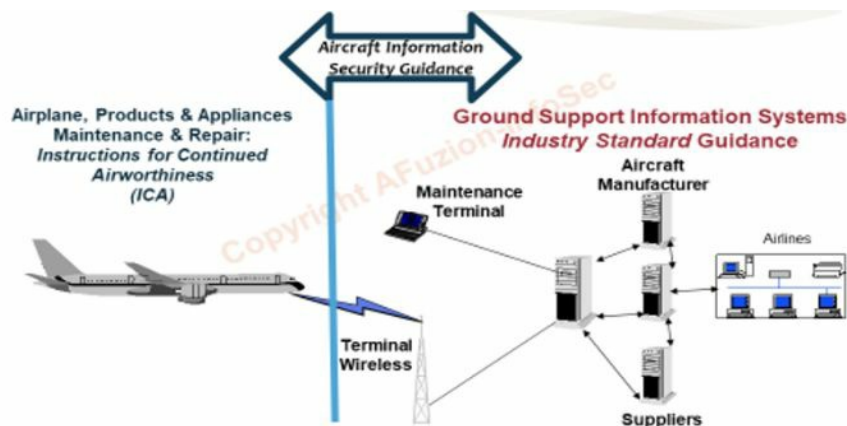
- a. AWSP comprises 7 steps: Plan for Security Aspects of Certification, Security Scope Definition, Security Risk Assessment, Risk Acceptability Determination, Security Development, Security Effectiveness Assurance and Communication of evidences.
- b. These 7 steps are detailed into 14 activities: Plan for Security Aspects of Certification (PSecAC), Plan for Security Aspects of Certification Summary (PSecAC Summary), Aircraft Security Scope Definition (ASSD), Preliminary Aircraft Security Risk Assessment (PASRA), Aircraft Security Risk Assessment (ASRA), System Security Scope Definition (SSSD), Preliminary System Security Risk Assessment (PSSRA), System Security Risk Assessment (SSRA), Aircraft Security Architecture and Measures (ASAM), Aircraft Security Operator Guidance (ASOG), Aircraft Security Verification (ASV), System Security Architecture and Measures (SSAM), System Security Integrator Guidance (SSIG), System Security Verification (SSV).
- c. These 14 activities include 62 objectives (combined) to be met.
- d. Detailed “Acceptable Means of Compliance” (AMC) – provided mostly

in DO-356A/ED-203A.



2) “Guidance for Continuing Airworthiness”, mostly provided by DO-355/ED-204, but supported by the “core” documents of the set as well. This Guidance is provided for the following eleven aspects: Airborne Software handling, Aircraft Components handling, Aircraft Network Access Points, Ground Support Equipment (GSE), Ground Support Information Systems (GSIS), Digital Certificates, Aircraft Information Security Incident Management, Operator Aircraft Information Security Program, Operator Organization Risk Assessment, Operator Personnel Roles & Responsibilities, and Operator Personnel Training.

Combined, the AWSP and Guidance for Continuing Airworthiness form one integrated entity that comprises a complete set of information-security AMC for achieving and maintaining airworthiness for the entire aircraft life cycle.



What Does It Take To Meet The “326/202 Set”

“Guidance/Recommendations”?

While DO-326A/ED-202A specifies the top-level formal requirements of the information-security airworthiness process, DO-355/ED-204 specifies in more detail what it takes to retain airworthiness, but here, too – the emphasis is on due process and following a variety of other, more elaborate information security generic standards, with clear roles for the Design Approval Holder (DAH) – typically the developer of the equipment, and the operator of the equipment.

As for the airworthiness certification process itself, it is mostly DO-356A/ED-203A that comes to the rescue. This 370 page long document details the specifics of the security aspects of the 3 key sub-processes of avionics certification: Planning (including Security Scope), Development, and Integral Process (including Risk Assessment and Security Effectiveness Assurance). Note that these three key sub-processes are thematic to many of the DO-XXX documents including DO-178, DO-254, DO-278, etc. as depicted for cyber-security below:



For the information security aspects, a more useful grouping of sub-processes would be: (1) Security Scope & Risk Assessment, (2) Security Development, (3) Security Effectiveness Assurance, while planning would refer to the top-level-certification.

These 3 sub-processes are further detailed in DO-356A/ED-203A (and partly in DO-326A/ED-202A):

1. Security Scope & Risk Assessment:
 - a. Security Scope Definition
 - b. Threat Conditions Identification & Evaluation
 - c. Threat Scenario Characterization

- d. Level of Threat Evaluation
2. Security Development:
 - a. Aircraft Security Architecture & Measures development
 - b. System Security Architecture & Measures development, which can be further broken down to the Sub-System level, Item level, etc.
 - c. System Security Integrator Guidance development
 - d. Aircraft Security Operator Guidance development
 3. Security Effectiveness Assurance – 118 Activities, in 39 Objectives, included in 13 Sections, of 2 types:
 - a. Security Specific Assurance – comprising 62 Activities, in 15 Objectives, included in 5 Sections
 - b. Security Development Assurance – comprising 56 Activities, in 24 Objectives, included in 8 Sections



DO-356A / ED203A Security Assurance: 39 Objectives 118 Activities

A crucial aspect of information security is “Time”: it is not merely a one-way safety-type process that has a clear end-point and then retreats to just monitoring – but a continued struggle against any potential attacks and attackers, that keep evolving even if nothing else happens concerning the said system. Thus, any modification, addition, removal or even altered function should be re-processed according to the DO-326/ED-202 set in order to assess what time has wrought. Furthermore, even the mere passage of time, without any modification of the system or its functions can bring changes in the hostility of the cyber-environment, so keeping up with the DO-326/ED-202 set would require periodic risk analysis in order to assess whether the system is still as secure as originally intended. Both the DO-326/ED-202 & DO-356/DO-203 “core” development phase documents and the DO-355/ED-

204 “continued airworthiness” phase document need to be followed on this aspect.

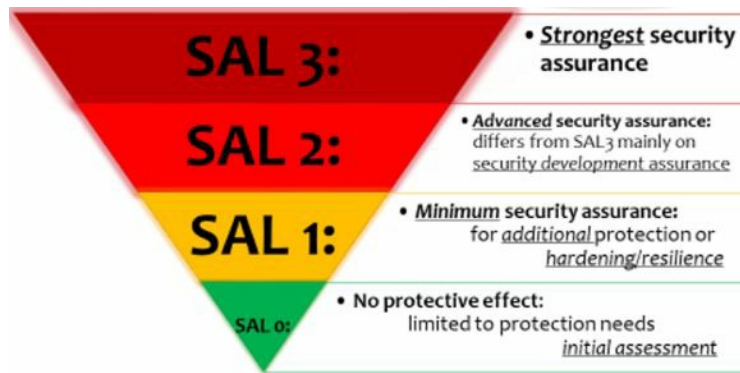
How Can The “DO-326/ED-202 Set” “Guidance/Recommendations” Be Efficiently Met?

Detailing the information security processes, even to the finest item does not ensure success, of course. Moreover – it definitely does not even start to deal with efficiency, in fact – DO-326A/ED-202A clearly states: “... the notion of efficiency, defined as the relationship between obtained results and resources engaged, is not considered in this standard.”, no less...

So, what should an applicant do to keep the time and cost of this process reasonable?

For Security Effectiveness Assurance, DO-356A/ED-203A specifies two methods to contain the required effort:

1. Up to ~25% of assurance-activities are not “security specific”, so they can be satisfied by providing evidence that the applicant has been applying “regular” safety development assurance practices, such as ED-79A/ARP4754A, DO-178C/ED-12C, DO-254/ED-80 etc.
2. The security measures requirements vary According to the Risk Level of the system, which mandates different levels of assurance. The levels of assurance assigned to the proper security measures are SAL: Security Assurance Level, similar in mentality to the FDAL/IDAL of the parallel safety assurance process. SAL 3 is the highest level of assurance, SAL 0 means – no assurance; just prove this is actually SAL 0. As a result, properly performing the Risk Analysis and designating appropriate assurance levels would mean that assurance efforts would be more relaxed for systems/items that are not rated SAL 3. The Security Levels are summarized in the following figure:



Security Assurance Level (SAL) for Security Measures

For the risk assessment and security development, there are various techniques, and even DO-356A/ED-203A provides, for instance, four different options for risk assessment that could be acceptable. The general approach of the DO-326/ED-202 set is – “there could be more than one acceptable means of compliance (AMC)”, so the applicant should carefully define the proper AMC for the specifics of their case. Even more so – the DO-326/ED-202 set recognizes that Airworthiness Information Security is “Work-In-Progress”, so that emerging measures, techniques, standards, even methodologies can be brought into play to streamline the process.

A final consideration that can considerably affect the efficiency of the Airworthiness Security Process is its integration level with the parallel Airworthiness Safety Process. One aspect of this similarity is – the two processes are very similar, so inputs can be taken in and lessons can be learnt from the safety process, so efforts are reduced. The other aspect that has to be considered is – the extent to which the processes are combined. The DO-326/ED-202 set's take on it is “it depends”: there are advantages to closely coupled safety and security, e.g.: avoiding the repetition of tasks and lowering the integration effort, BUT, there are also prices to such a close coupling, e.g.: whereas safety requires as few changes as practical after configuration change, security may require very frequent changes to accommodate changing threats, so the safety and security aspects may contradict each other if integrated into the same item.

Summary & Conclusion: Paranoids Live Longer

A passenger walks into a commercial-flight airplane with a laptop, hacks its network, making it fly even higher: extremely unlikely when using the DO-

326/ED-202 set. The major takeaways following this brief introductory review:

1. Aviation Cyber-Security...
 - a. ...is a discipline of its own – it needs unique expertise, and requires its own analysis techniques and assurance considerations;
 - b. ...involves many stakeholders, internal and external to applicant's organization, with whom All-Way Trustworthiness should be established – not “Assumed”...;
 - c. ...is a “never ending story”, as cyber threats keep evolving with time – so Airworthiness Security needs to be implemented, as an ongoing process throughout the entire lifetime of all certified aircraft and other equipment.
2. The DO-326/ED-202 set...
 - a. ...is already a usable set of documents, regarded as the Acceptable Means of Compliance (AMC) for Aviation Cyber-Security in the U.S. and Europe, and rapidly becoming so in the rest of the world;
 - b. ...should be approached as a whole, rather than “document-by-document”;
 - c. ...draws from a variety of solid references in the areas of Cyber-Security and Aviation-Safety to create an entire “eco-system”, to the extent that certain DO-326/ED-202 set methods and solutions are derived directly from these references – thus, providing multiple specific options, that need to be applied as a function of the specific organization, project and system;
 - d. ...is still an evolving “Work In Progress”, so care should be taken to not “set in stone” any specific solutions.
3. The Airworthiness Security Process (AWSP)...
 - a. ...is the core process of the DO-326/ED-202 set, from which all the set's documents draw;
 - b. ...includes 7 steps, 14 activities, 62 objectives – so cannot be treated as an “afterthought” of airworthiness certification;
 - c. ...heavily relies on integral processes, namely “Security Effectiveness Assurance”, including 118 objectives in 39 activities – applied as a function of a few variables, mainly the “Security

Assurance Level” that determines the required security defense level of the element under consideration;

- d. ...is similar in nature to the Airworthiness Safety Process, as both use hazard/risk assessments, severity of failure/threat, mitigation requirements and similar assurance techniques – and this similarity provides rich opportunities for mutual benefits for the two processes, including the usage of up to ~25% of Safety Assurance evidence as Security Development Assurance evidence...
- e. ...however, the most cost-effective level of integration between the Security and Safety Airworthiness Processes depends on a variety of specific variables – organizational, equipment type etc. – so it should be best determined on a case-by-case basis.

As these points amount to 12 major takeaways, it is necessary to add one more takeaway, to get to exactly 13, so it would be appropriate to conclude with a sound advice from Mr. Pedro Bustamante, Technology VP, Malwarebytes, which may arguably be the most important takeaway:

“If you’re not paranoid, you’re not going to survive.”

Engineering Transitions

By Vance Hilderman

Reviewed by:

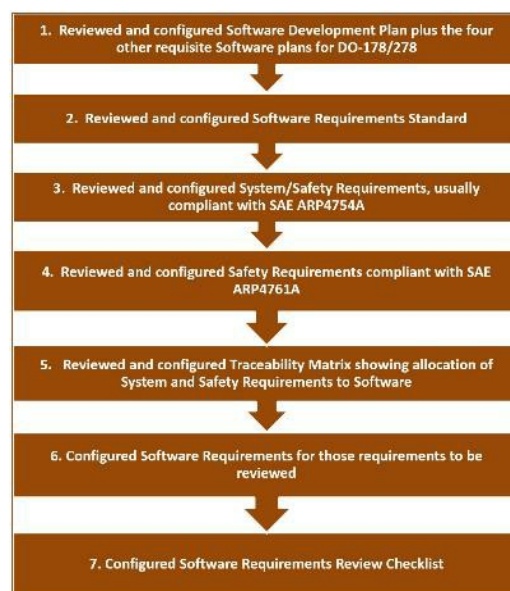
- Mr. Kevin Kendryna, Quality Engineering Leader, General Atomics
- Ms. Lisa Wynn (Duncan), Software Quality Engineer, M.A., CSQE.

When safety is critical, safety-critical systems must ensure the orderly, measured conduct of engineering activities and sustain attention to detail. Quite literally every engineering activity has entry and exit criteria which must be defined and documented in advance with a corresponding set of product and process criteria (that will be used by a certification authority) to assess and ensure adherence. For example, before implementing software logic (“coding”), critical systems must have various pre-defined, formally reviewed artifacts in place (under formal configuration control) including safety/system/software requirements, design data, and software coding standards by which the code can be verified. The order of these engineering activities is thus predicated on “transitions” (mini gate-reviews) which must be assessed by QA and sometimes approved by the certification authority (FAA, EASA, Military, etc.) before the development team can proceed to the next phase.

These engineering transitions can be defined in a separate document or embedded within the applicable Software Transition Plan (STP) as an additional process document which defines the various engineering lifecycle transition steps that are performed during planning, development, and V&V. While defined transitions are required per aviation engineering guidelines such as DO-178 (airborne software), DO-254 (airborne hardware), DO-278 (ground/space- based Communication Navigation Systems / Air Traffic Management – CNS/ATM), many companies embed them within the Software Development Plan (SDP) and the Software Verification Plan (SVP) instead of placing them in a separate STP document. Also, since

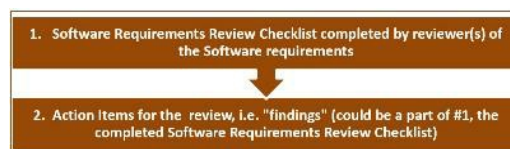
Quality/Process Assurance (QA/PA) is generally tasked with auditing and assessing adherence to transition criteria, the Software Quality Assurance Plan should detail such audits.

Transition planning is necessary to show that predefined entry and exit criteria exist for engineering transitions, and that those [entry/exit criteria](#) are followed and audited. For example, software requirements must be reviewed and baselined prior to initiating the high-level software design. The following items, i.e. “entry criteria”, must be present prior to the associated software requirements review:



Sample: Entry Criteria Preceding Software Requirements Review (“Transition”)

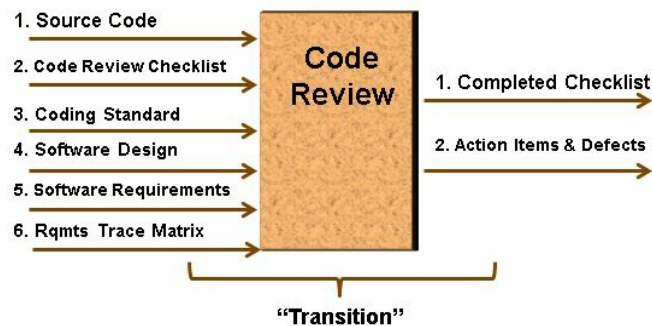
Only when the above seven items are available can the associated Software Requirements Review be initiated. Upon completion of the Software Requirements Review, the following items, e.g. “exit criteria” will be completed:



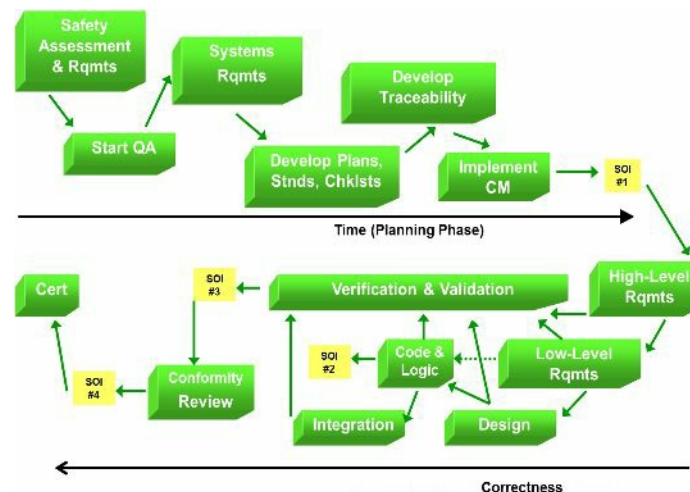
Sample: Exit Criteria Following Software Requirements Review (“Transition”)

As an additional example, consider an official for-credit software code review or hardware logic review. For a code review which adheres to DO-178 or DO-278, the following diagram shows the entry and exit artifacts for

that review and each of these must be under configuration management:



The above Entry and Exit artifacts thus form one “transition”. This transition process is repeated for all of the DO-178/278 engineering transitions, albeit with different entry and exit criteria defined for each such transition. Now consider the various transitions within a DO-178/278 optimal engineering route; each arrow below constitutes a “transition”:



Optimal Engineering Route, with Transitions, per Vance Hilderman

In the preceding figure, each of the key engineering activities is represented by a box and the arrows representing ‘transitions’ connect the boxes from start to finish. Each of these arrows has unique, pre-defined entry tasks, verification, and exit criteria associated with it and those criteria must be satisfied in order for the activity to be deemed appropriately performed. But wait you say: just because the process was followed and transition criteria satisfied, how does that prove product perfection? It doesn’t. That is not the purpose of process-based transition criteria. Assessing transitions is about “process”: ensuring advance consideration and formal definitions of

entry/exit criteria, then assessing the degree to which engineering followed. Remember: the verification engineer should use the entry/exit transition artifacts in performing their verification activity; the Quality/Process Assurance personnel then assess that engineer to ensure such transition criteria were followed with evidence thereof. Remember:, there is no such thing as ephemeral product perfection, even in aviation (rather, the ecosystem of safety assessments ensures such imperfections meet acceptable risk factors). Second, engineering reviews are meant to assess product technical attributes while QA conducts transition assessments in order to provide the certification authority with independent objective evidence of process-based compliance aviation guidelines such as DO-178, DO-254, DO-278, et al.

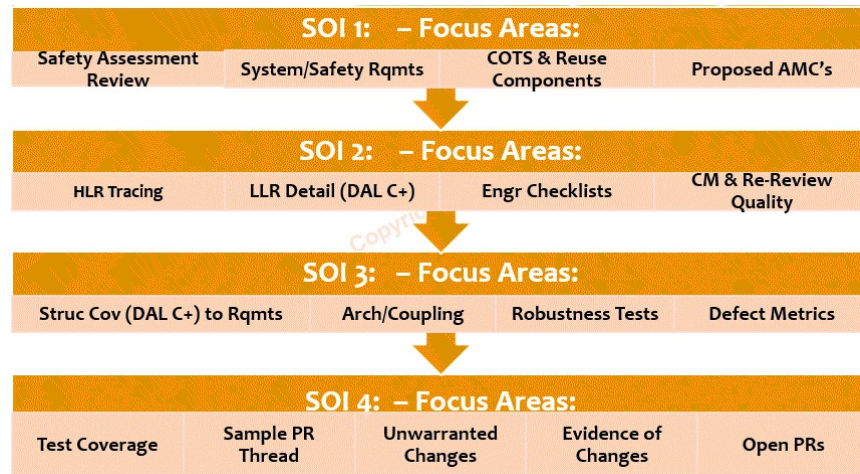
Stages of Involvement: “SOI’s”

In the figure above, note that there are four SOI’s, “Stage of Involvement” activities commonly applied to aviation engineering. These SOI’s comprise four major “gates” to assess top-level transition attainment. SOI’s are process-based and generally led by QA/PA. The purpose of each SOI is depicted below:



Synopsis: Essential Purpose of Each Stage-Of-Involvement (SOI)

For a more detailed view of the SOI’s, the following figure summarizes key activities associated with each SOI:



Mid-Level View of Each SOI

SOI-1 Details

SOI-1 is similar to a review of building plans prior to constructing an office building near an earthquake fault-line: have you properly evaluated safety considerations to ensure the building will be safe if the plans are followed? In aviation, SOI-1 includes the following key questions to ensure answers are clear and verifiable:

- Has the Functional Hazard Assessment (FHA), Preliminary System (or Aircraft) Safety Assessment (PSSA/PASA), and Common Cause Analysis been initiated such that risks are identified and the proposed development assurance level (DAL) is justifiable?
- Have the means to comply with ARP4754A for the Aircraft or System, and ARP4761A for Safety, been identified along with initial safety and aircraft or system requirements?
- Are the requisite plans and standards all completed in compliance with the applicable and defined regulations?
- Has the aircraft or system architecture been defined including use of COTS and legacy items?
- Are there any alternative means of compliance (AMC) proposed and is there justification for their acceptability?
- What engineering tools are planned to be used and is any need for qualification addressed?

- If this SOI-1 is approved, would it be acceptably safe to initiate aircraft/system development following these plans?

SOI-1 thus provides a “transition” to development which would then itself culminate with SOI-2.

SOI-2 Details

SOI-2 is similar to a building inspection when major portions of the building are complete: did the construction meet the previously approved (e.g. SOI-1) building plans? SOI-2 includes the following key questions to ensure answers are clear and verifiable:

- Did development of the requirements, design, and code/logic follow the previously approved (SOI-1) plans and standards and transition criteria based upon audited verification evidence?
- Are requirements (System/Safety requirements, software high and low-level requirements, hardware requirements) present with bi-directional traceability and verification records affirming process-based decomposition and refinement?
- Are review and audit records present to show requirements, design, and code/logic were developed according to the pre-defined transitions described earlier including review checklists for each?
- Do configuration management (CM) records exist which show artifact controls were in place ensuring all transition inputs/outputs were captured, all changes after initial reviews were accompanied by recorded re-reviews?
- Were QA/PA activities performed independently, and were corresponding verification activities for DAL A and B artifacts performed independently as required?

SOI-2 thus provides a “transition” to verification which would then itself culminate with SOI-3.

SOI-3 Details

SOI-3 establishes that the validation and verification activities were

completed according to plans and standards. SOI-3 includes the following key questions to ensure answers are clear and verifiable:

- Have reviews and tests been applied to all the software according to the Development Assurance Level (DAL), including traceability, transition criteria, and structural coverage assessment (code and requirements) for DAL C, B, and A?
- Is the architecture/design deterministic as verified via requirements-based tests and data flow / control flow assessment and coupling analysis?
- Has robustness testing been sufficiently performed and reviewed?
- Have defects discovered during the verification process been dispositioned?
- Optionally: have defect metrics been analyzed to determine sufficiency of related processes?

SOI-3 thus provides a “transition” to Conformity which would then itself culminate with SOI-4.

SOI-4 Details

SOI-4 establishes that prior processes (SOI 1, 2, and 3) have been performed throughout the engineering lifecycle in compliance with the stated criteria and the validation and verification activities were completed according to plans and standards. SOI-4 includes the following key questions to ensure answers are clear and verifiable:

- Have all changes to plans/standards/checklists been approved and documented including justification for items not conforming to the latest revision of such?
- Have all changes been performed following the specified configuration management processes?
- Have all unresolved problems been subjected to safety analysis which detailed potential safety impact and operational constraints.
- Is there sufficient evidence of requisite quality/process assurance audits

with confirmation of discrepancy resolution?

- Is the configuration index complete such that future artifact assessment, changes and reverification can be performed on the operation item including recreation of an identical item during the operational lifetime of that item?
- Is there a statement of compliance to applicable standards/guidelines in an Accomplishment Summary?

Transitions thus perform a cornerstone to foundational aviation development certification. While it is most important to plan for, and document, the detailed transition for the engineering processes, it is equally important to independently assess such transitions via Quality Assurance and keep records thereof. This author has soon far too many otherwise good aviation projects fail their certification authority transition assessment and have to spend inordinate resources going back to the starting line.

Aviation Development Traceability

By Vance Hilderman

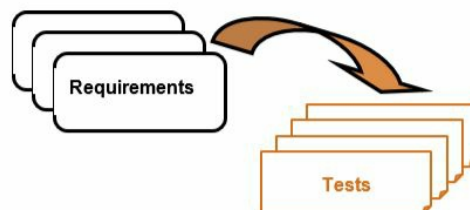
Reviewed by:

- Mr. Bill De Leeuw, FAA Designee, Systems and Equipment, Software
- Mr. Kevin Kendryna, Quality Engineering Leader, General Atomics
- Ms. Lisa Wynn (Duncan), Software Quality Engineer, M.A., CSQE.

Aviation Development Traceability: The Cornerstone.

Traceability is a key aspect of most modern professions. Accountants, researchers, and forensic scientists alike all apply traceability. For high quality and safety-critical engineering development efforts however, traceability is a cornerstone not just for achieving success, but to proving it as well via Certification.

Most system, software, and now hardware, engineering standards require varying degrees of traceability. Why? Traceability is a readily understood means of ensuring that implementation corresponds to specification. Traditionally, traceability has meant simply ensuring that each unique requirement has corresponding test cases which verify the requirement was implemented. In such a simplistic application of traceability, one need merely develop tests to cover each requirement then associate, or “trace” those tests to that requirement. When all requirements were thus “traced” to specific test cases, traceability, and likely the testing, was deemed complete as in the following figure:



Simplistic Traceability - The Beginning

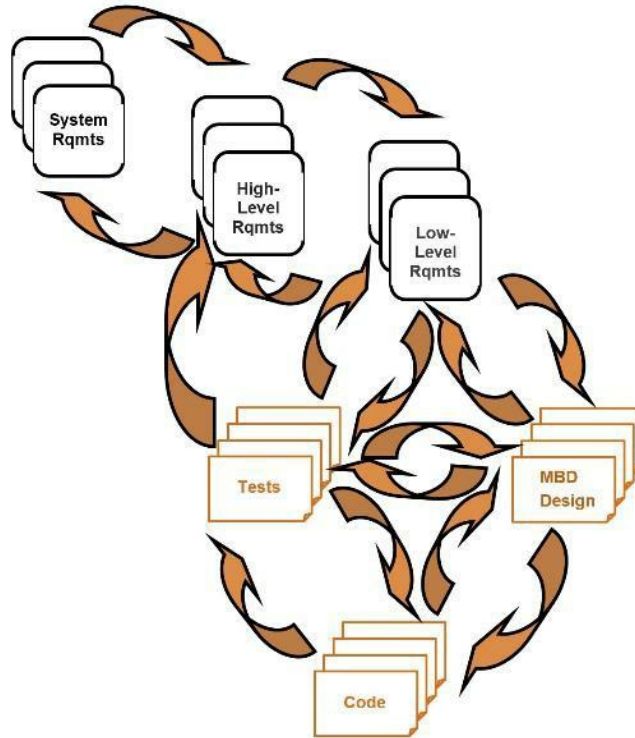
The aforementioned simplistic approach to traceability indeed has merit, and it is a first step in the right direction. But is it complete? Hardly! Does it satisfy DO-178C or DO-254, arguably the world's most stringent standards as required for avionics systems? Absolutely not! What are the shortcomings with this simplified approach to traceability?

- *There is no traceability to the implementation components; necessary during the review of such components*
- *There is no traceability between increasing levels of requirement granularity; necessary to prove the requirements decomposition process*
- *There is no traceability to design data; necessary for review of such*
- *There is no ability to isolate extraneous implementation components, such as unused code*
- *Traceability is one-way, not bi-directional*

To remedy these shortcomings, DO-XXX and ARP47XX require traceability which is more detailed, more complete, and which traces not only forwards but backwards as well, e.g. “bi-directional”. And DO-178C, DO-278A, and DO-254 requires more complete traceability throughout a complete set of artifact components including:

- *System Requirements*
- *High Level and Low Level Software or Hardware Requirements*
- *Implementation (Code or VHDL)*
- *Tests*
- *With the above captured in the Configuration Index (CI)*

This more complete traceability, applicable to just software per DO-178C and DO-278A, is depicted in the figure below (note that “MBD Design” could be replaced with non-Model-Based Design and thus simply become “Design” as appropriate):



Modern Aviation Traceability - The Result of Evolution (with MBD Depicted)

As can be seen from the above Figure, DO-XXX & ARP47XX traceability essentially forms a closed circuit and is bi-directional. It is closed-circuit because traceability is top-to-bottom via requirements to code and tests, as well as bottom-to-top, from code and tests, to the associated requirements. Also, traceability should provide for one-to-many and many-to-one associations. What does this mean? Simple: each traceability item can trace to multiple lower-level items, while multiple higher-level items can trace to the same lower-level item. For example, a single System level requirement can (and likely will) be decomposed into multiple lower level requirements; thus that System requirement traces to each of those lower-level requirements. Conversely, a software requirement may need several test cases to verify it; each of those test cases thus traces up to that single software requirement to provide aggregate coverage of that requirement.

Within safety-critical systems, “Safety” is an obvious priority. Typically safety related requirements are identified via the safety analysis processes governed by ARP-4761A and ARP-4754A. Those requirements are paramount and must be considered throughout the engineering and subsequent operational lifecycles. Traceability of those requirements to lower

level requirements which assist in or relate to safety implementation is required. Whenever an activity (change, verification, etc) is performed on a requirement related to such a safety requirement, a feedback to the safety process must be activated; traceability is the key mechanism to invoke and denote such safety relationships.

Traditional industry traceability, not DO-XXX, is one way, from top to bottom, which ensures that each requirement is implemented and verified. In many industries this top-to-bottom traceability is deemed sufficient. However, in safety critical industries, and for DO-XXX in particular, such one-way, top-to-bottom traceability is insufficient. Why? Because high quality systems must ensure that there is no extraneous implementation or testing. For example, unused and undocumented code is the bane of product maintenance and quality assurance because such unused code can lead to misunderstanding and incorrect assumptions about the code behavior, particularly during code reviews or product upgrades. This ultimately increases the probability of run-time errors. So, DO-XXX for hardware and software require that all code functions have at least one requirement tracing to them, in order to ensure that all functions have a reason to be present in the end-product. In addition, a configured traceability matrix is required to be used as an input to reviews; for example, a software code review requires consideration of the software requirements allocated to the code being reviewed, per the configured (controlled) traceability matrix.

Why Traceability?

If you asked ten engineers for the most significant contribution provided by traceability, you would likely receive ten different answers. In fact, traceability provides many different benefits. The Systems Engineer would state that traceability was used to ensure that all the specified requirements were implemented. The Designer would state that traceability ensured a proper architectural allocation of requirements via an optimal design. The Coder might suggest that traceability helped ensure capture of all the functional requirements within the code. The Tester would state that traceability assists with correlating each requirement to the test case(s) which verify that requirement. The Quality Assurance person would state that traceability provides the ability to ensure that all the other engineering

disciplines have done their job. Finally, the Engineering Management would state that Traceability metrics enable accurate project statusing to avoid the “I’m 95% done boss, trust me” response to project status.

Therefore, with modern software and hardware engineering processes, traceability is used to answer the following questions, at a minimum:

1. *Where is each requirement implemented?*
2. *Is every requirement allocated?*
3. *Is the implementation compliant with the requirements?*
4. *Is the requirement related to Safety, and if so, are traces to related and parent Safety requirements clearly denoted for continual safety analysis impact?*
5. *What verification test cases will be used to verify a given requirement?*
6. *What is the aggregate impact of changing any requirement or code?*
7. *Is the project complete and ready for certification?*
8. *Is the deliverable product indeed what was originally requested and specified?*

The last item above is interesting: an implicit additional important purpose of traceability is to prove that the configured product being delivered to the customer is indeed the product that was requested, via the configuration index and all of the configuration management activities carried out to ensure all of this traceability is maintained. There are many different types of traceability systems and tools; however they must all provide the ready ability to answer the above questions at a minimum in order to be deemed sufficient to meet the certification requirements of DO-XXX and ARP47XX.

Traceability Tools

It’s now understood that complete traceability is not only desired for safety critical systems but is required. However, are dedicated traceability tools also required? No; modern tools are not required; just as modern tools are not required to build a house. Some people in the world still have little choice but to live in houses they built out of sticks and mud. But in the real-world, e.g.

your world of engineering development, traceability tools are necessary to improve accuracy, repeatability, and schedule attainment. Ultimately, traceability needs to be proven to be both accurate and complete.

For traceability, the end-result requires an ability to provide traceability data between key project components including requirements, design, implementation, and tests. Theoretically, and for very small projects, it is possible to provide traceability via a spreadsheet or simple table within a document. Generation and maintenance of such spreadsheets and tables would be manually intensive and prone to errors; specifically manual traceability does not readily provide for updates or expedient quality assurance audits. However, for very small projects manual traceability can be suitable. For somewhat larger projects (greater than 2,000 - 3,000 lines of code or greater than 100 requirements), a traceability tool will always prove cost-effective, saving at least 5X\$ over manual traceability. But what options exist for traceability tools?

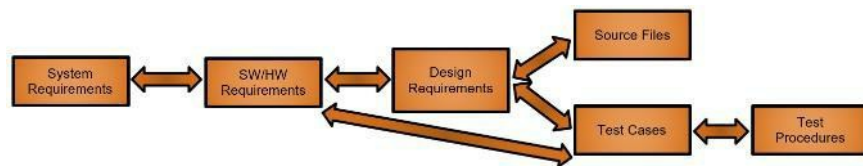
- a. Build your own, in-house; or
- b. Buy a dedicated traceability tool, one designed for DO-178C/254; or
- c. Buy a COTS traceability tool such as DOORS, RequisitePro, Jama, JIRA, etc.

What are the basic attributes of an acceptable traceability tool? Regardless of your tool choice, you should ensure your tool provides for:

1. Flexible tagging mechanisms, so your standards do not require change.
2. Bi-directional and one-to-many-and-many-to-one traceability between requirements, design, code, and test elements.
3. Automatic updates to avoid error-prone manual tracing.
4. Automatic indication of missing trace elements.
5. Tool-specific ability to interface with a requirements management tool for downstream integration.
6. If you are building avionics, compatibility with DO-XXX and ARP47XX requirements.

Do traceability tools need to be formally qualified per DO-330 (as covered in a prior chapter in this book)? Such would appear so at first glance because they automate traceability aspects required by DO-XXX. However, if the outputs of the traceability tool are manually verified, qualification is not required. Since most traceability tool users do perform a manual verification of traceability as part of Stage of Involvement (SOI) #4, qualification of the traceability is not required or performed.

As a final check, be sure your traceability tool can support a typically basic traceability paradigm such as that depicted in the figure below for software; note that hardware also needs traceability and ideally would be supported by a common tool/database.



Basic Systems, Software, and Hardware Traceability Structure Paradigm

Aviation System Validation & Verification

By Vance Hilderman

Reviewed by:

- Kenneth R. Hebert, PhD, Senior Systems & Process Subject Matter Expert, AFuzion Inc.

Validation & Verification, Verification & Validation, or just “V&V”?

Validation & Verification (V&V) are cornerstones of aviation and safety-critical software development. Actually, V&V are key ingredients of any successful engineering development framework. If an entity has not been validated or verified, how does one know what that entity really constitutes and can it be relied upon? Many publications and industries refer to V&V as “Verification & Validation”, placing the “Verification” first. However, any good engineer knows that validation both initially precedes verification and is more important than verification. Validation provides insights into a product’s capability while verification assesses its contents. Why then do DO-178C and DO-278A call for verification but not validation? More on that in a few paragraphs ...

Verification and validation provide the foundation of aviation’s “correctness” process. For the simplest definition, the following applies:

- **Validation:** *Assessing the degree to which an entity’s associated requirements are correct, unambiguous, complete, and verifiable.*
- **Verification:** *Assessing the degree to which an entity’s implementation meets its requirements.*

Validation occurs over the engineering lifecycle but initially precedes design, so that requirements can be assessed and improved prior to design and implementation; this ensures that subsequent design and implementation can be reviewed according to the associated requirements. Conversely, verification occurs after an artifact or item has been constructed so an

assessment can be made as to its correctness. Proper validation prior to implementation can prevent defects. Verification can never prevent those defects but instead detects them such that subsequent corrections can be applied. Researchers generally agree that preventing errors (including validation) is 500% - 1000% more cost-effective than detecting those errors via verification. Therefore, validation both initially precedes verification and is more cost-effective. Why then do the aviation software guidelines DO-178C and DO-278A seemingly ignore validation and instead focus on verification? Because validation includes assessment of requirement completeness which cannot be performed fully until the hardware and system are also integrated; until then, validation is subjective at the software level. But review of software requirements is vital and the foundation of DO-178C and DO-278A; this requirement review is performed as a verification activity prior to software design and thus replicates a portion of validation.

Initial validation begins by assessing an entity's requirements and continues through to implementation. In aviation, validation is performed upon the aircraft, hardware, and the system; but not the software within a system. Why? Because of the validation definition provided above. Validation includes determination of requirement "completeness". But software alone cannot be validated because such validation requires completely integrated hardware and software. Therefore software "validation" is inherently performed as part of the aviation system V&V. Validation must cover both intended, and unintended, functionality. It is performed by applying a combination of the following techniques. to ascertain if an entity is correct, unambiguous, complete, and verifiable:

- Bi-directional Traceability
- Analysis
- Modeling (see subsequent section)
- Test
- Similarity
- Engineering Review

Verification meanwhile focusses upon evaluating an artifact for adherence to

its requirements. The “verification equation” of aviation software is depicted below:



Verification Equation Depiction

Reviews, Tests, and Analysis are summarized below.

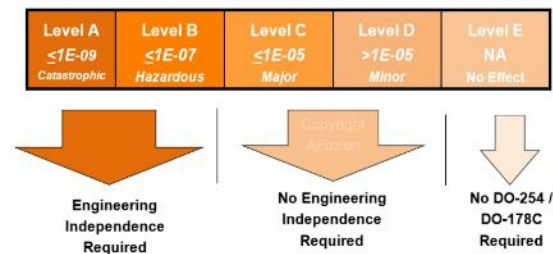
Reviews .

As shown above in the verification equation depiction, “Reviews” (largest font) depict the largest set of work as all artifacts developed by aviation engineers in support of certification are reviewed. Specifically, the following key aviation artifacts are reviewed in the case of aviation software:

- Plans & Standards
- Aircraft, System, Software, and Hardware Requirements, plus corresponding:
- Design
- Code
- Tests/Results
- Traceability
- Problem Reports/Corrections

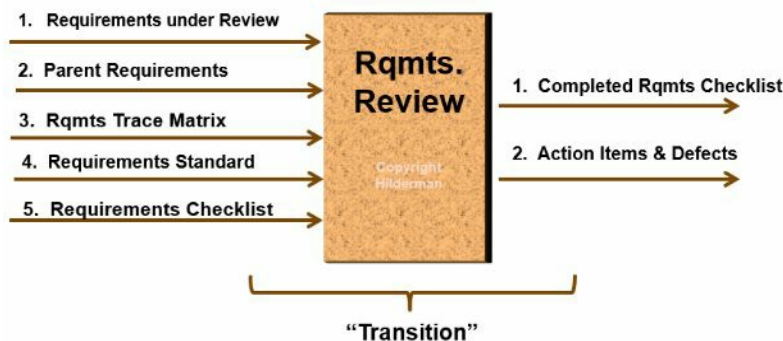
Reviews need to be performed independently for more critical artifacts as determined by the Development Assurance Level (DAL). Typically, DAL A and B process outputs (e.g. “artifacts” such as requirements, design, code, tests, etc.) require proof of independent review. The following diagram depicts the application of V&V independence based upon varying system criticalities for large aircraft/systems:

Engineering “Independence” per Development Assurance Level (DAL). (Note Quality Assurance must always be independent):



In the above figure, “Independence” means a different person applying an uncoupled process. Clearly individuals are independent. However individuals can be related in subtle ways. For example, if John is to be an independent reviewer of Mary’s GPS system development, John is not allowed to rely upon Mary’s potentially related (and thus faulty) GPS simulator. One key independence exception is hardware design verification: Since hardware design is uniquely iterative and complex for many reasons including thermal, resource, and layout considerations, using an independent person to develop tests can actually provide less thorough verification. Therefore for hardware, unlike DAL A and B software, the designer is allowed to test their own design provided an independent person then reviews those tests.

Aviation Reviews need to be performed with consistency and evidence. Therefore, reviews prescribe a predefined set of inputs and outputs specific to the type of item being reviewed. For example, hardware and software requirements require a requirements review for all safety-related DALs (which would be DAL A, B, C and D but not E). In the case of software requirements, the following predefined inputs and outputs are applied which comprise “transition criteria” (See Chapter 17 herein for Transition details).



In the above figure, the requirements reviewer(s) uses all five inputs to perform the review. Any noted action items or defects are then fed back

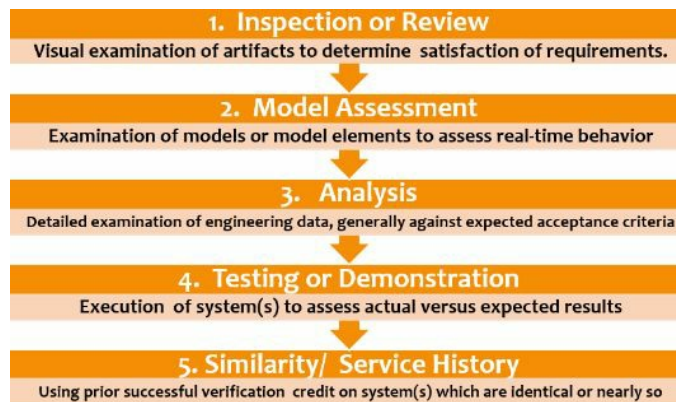
through the requirements development process and the review is repeated to ensure correct updating. Requirement reviews often yield updates to parent requirements for clarification and also assess the correctness of mandatory traceability to parent requirements, so synchronization between inputs and outputs is paramount and all tracked as part of the verification and configuration management processes. Quality Assurance then audits those reviews to ensure record-keeping supports proof that the inputs and outputs were properly applied during the review process; this is termed a transition criteria audit. All other artifact reviews noted previously likewise require transition criteria with record-keeping thereof. See Appendix B of this book for a sample Software Requirements Review checklist.

As with all things Aviation, V&V requires defined planning prior to initiating the actual activities so that subsequent reviews of those activities can be assessed for consistency to that prior planning. In order for V&V to provide the confidence needed, myriad details need to be accounted for and innumerable decisions have to be made which balance cost and insight gained. V&V can be complex and intricate enough to demand that it be systematically planned (as would any complex and resource intensive activity). At a minimum, the following information should be addressed (via objective assessment criteria) in validation planning:



Validation Planning Topics

System-level verification (typically performed in aviation per ARP4754A) is summarized in the following graphic and further addressed below:



System-Level Verification Methods

Tests.

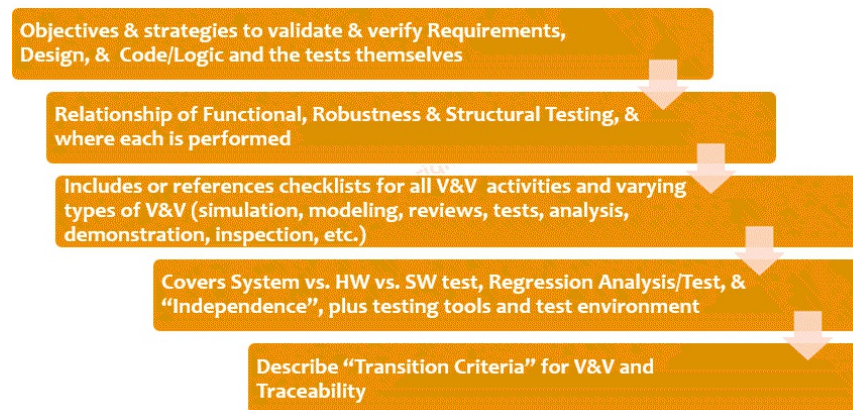
Testing involves actual execution of software and hardware, potentially within a target environment representative of an actual operational aviation system. The focus of testing should be requirements-based, but can also consider actual logic design and implementation. For example, more critical logic will additionally assess logic robustness and structural coverage analysis of that logic during requirements-based testing. In the earlier days of aviation systems testing, it was believed that testing directly improved quality and therefore testing of each hardware and software component was emphasized. And for software, it was believed that software is merely a collection of components and therefore thoroughly testing each component would improve software quality. The folly of those fallacies was soon apparent. As we've progressed we've come to realize that testing is simply the stimulus of an object in a prescribed environment and subsequent measurement of results. In any interesting situation, the variables involved make exhaustive testing impossible. By its very nature testing is a sampled activity from which general conclusions are drawn. The "art" of testing is to maximize information from the relatively small set of tests that can practically be run

Testing does not improve quality but instead provides a vital means of assessing quality; with a proper feedback loop testing can be used to understand and thus quantify potential shortcomings, with improved requirements, design and implementation to then address those shortcomings.

As software products have become more complex, testing has evolved.

Increasingly, software products are collections of components interacting in complex and subtle ways. But software is more than a collection of components because the manner in which those components interact is related to system quality. Therefore, as the DO-XXX's have evolved and improved, testing has been clarified to emphasize integration and system testing over component testing.

For aviation testing, the following information should be addressed (again via objective assessment criteria) in a corresponding Verification Plan:



Verification Planning Topics (Usually Merged with Validation Planning)

Aviation and avionics systems undergo system-level testing typically per ARP4754A, whereas the software and hardware within those systems are subjected to testing per DO-178C and DO-254 respectively. An exception is ground/satellite based systems which fall under DO-278A yet the hardware is not subjected to certification outside the system therefore DO-254 does not apply; this is because ground systems make ample use of COTS hardware and testing of those systems includes integrated hardware. At that System level, the verification method is typically allocated to one or more of the following four key techniques:

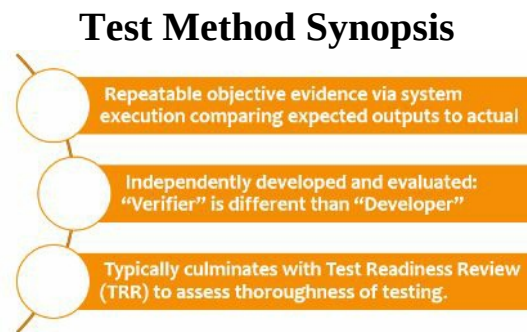
1. Test
2. Demonstration
3. Analysis
4. Inspection

While no accepted formula exists to specify which technique should be applied when or the relative priority of the four test methods above, it is this author's opinion that the relative priority should be as provided in the above

order.

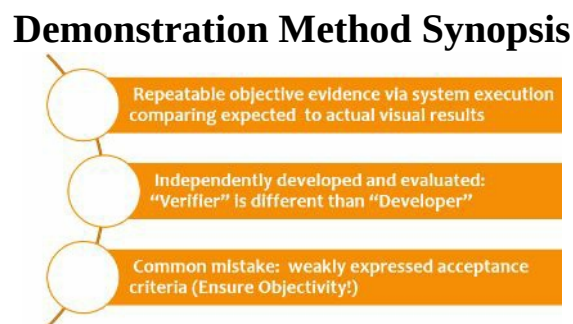
Method = Test:

Test involves actual execution to determine if actual results meet expected results and is the most preferred of the four major methods since it is less subjective, more repeatable, and most closely replicates actual aviation operations. The Test method is summarized in the following figure:



Method = Demonstration:

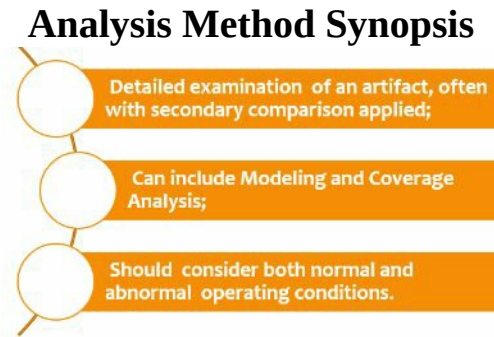
Demonstration involves actual execution where visual outputs (for example outputs visible on a cockpit or a ground controller display) are compared to expected visual results. Demonstration is typified by manual or semi-automated execution with a test operator in the loop who makes visual assessments. Since there is a operator or inspector, it is important that visual acceptance criteria be strongly expressed to minimize subjectivity. The Demonstration method is summarized in the following figure:



Method = Analysis:

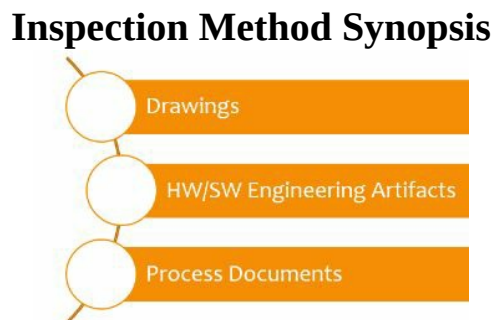
Analysis utilizes additional scientific or mathematical criteria applied via a defined process to assess the correctness of an artifact. Analysis is often a secondary comparison applied when Demonstration, Test, or Inspection

would not sufficiently assess results. Examples of analysis include traversing a bi-directional trace matrix seeking missing or incorrect traces would be a form of analysis as would assessing internal software model elements or code branches to determine if sufficient coverage were achieved. The Analysis method is summarized in the following figure:



Method = Inspection:

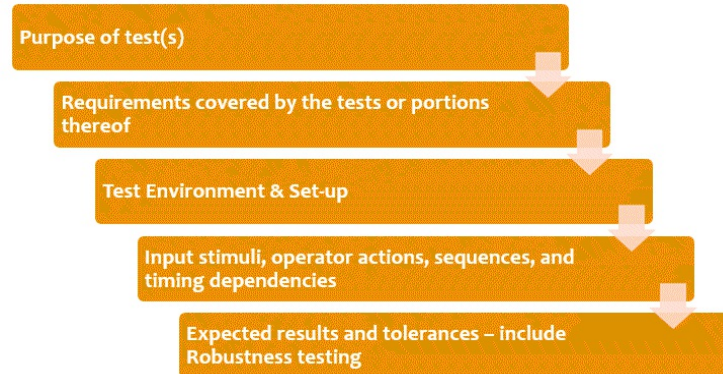
Inspection involves visual examination of an artifact which is not being dynamically executed and thus is “static”. The artifact is examined to see if specific features are present (or to confirm that features are not present). Often the feature list comes from characteristics that customers of the artifact have been found to desire or have had concerns with. For example, a piece of test equipment might be inspected to make sure it has a USB connection or that the connection does not come out when stressed. Plans and resulting engineering artifacts are often inspected as are calibration records. The scope of the Inspection method is summarized in the following figure:



Inspections such as Code Inspections (also called Code Reviews or Code Walkthroughs) are a specific form of inspection which are technically under the “Review” category explained previously). Aviation requires formalization of testing namely via a Plan which describes the overall testing approach as

described above, then also test Procedures and after executing the procedure, test Results.

Test Procedures should specify or reference the following information:



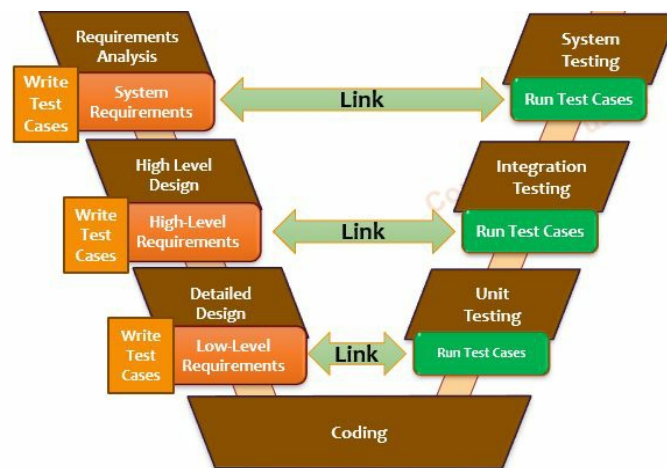
Test Results should specify or reference the following information:



Software & Hardware Logic Testing

In aviation, there are some forms of hardware items which are considered Simple Electronic Hardware (SEH) and thus require no special testing; they are simply documented, defined, configured, controlled and verified within the context of the system they are installed within. For example, resistors, capacitors and stand-alone discrete devices such as AND/OR gates require no additional dedicated testing but rather simple proof that defined system-level tests exercise them and would thus detect defects. These simple devices procured as Commercial Off-The-Shelf (COTS) are allowed to have presumed compliance to DAL A when properly selected and managed as described previously in this paragraph. However, silicon-based logic is almost always Complex, not Simple, because the user cannot prove they have evaluated that logic against all “foreseeable operating conditions”. Remember, with all but the most trivial asynchronous systems, the number of

foreseeable operating conditions approaches infinity; therefore, it's dysfunctional to believe one can provably verify an infinite number of input/output possibilities across the time domain in the absence of a formal mathematical model. (Even with such a formal mathematical model, mathematical closure must be proven per DO-333 in order to allowably reduce traditional testing activities described herein.) Therefore, aviation logic requires a defined evidence-based engineering lifecycle to compensate for its otherwise infinite test cases to prove correctness. Modern systems (regardless of whether they are hardware or software) integrated simple hardware or single code instructions into a more useful – but also more complex – package. While we can presume the single capacitor or the add instruction will work as expected, it is the logical structure that determines if they can be relied upon. That logic is tested at multiple levels as depicted in the following figure:



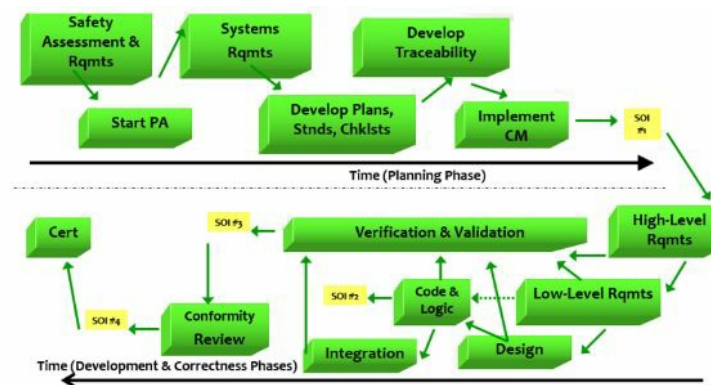
Typical Aviation V-Model Test Approach

Aviation guidelines are typified by the “V-Model” as depicted above since the timeframe of those guidelines’ development coincides with V-Model recognition. In the V-model, the aircraft requirements are first development, then the system requirements, then initial software/hardware requirements, then detailed software/hardware aspects, e.g. “Top-to-Bottom” representing the left hand side of the “V” in the above figure. Then, the testing is performed Bottom-to-Top beginning with hardware/software logic, then integrations, then the system and finally the aircraft.

For aviation systems containing logic, additional development and

verification rules apply since there is no way to otherwise ensure correctness of that logic. When the logic executes via form of microprocessor or microcontroller, it is termed “software” since the logic instructions are “soft” via their being first compiled from source code, then linked to build an executable, then loaded into a form of random Access Memory (RAM) to finally execute via a COTS processor containing a program counter and registers. Software is thus certified via DO-178C for airborne avionics and DO-278A for ground/space based systems. Aviation software is most commonly written in a high-order language (HOL) such as C, C++, or Ada, with Java increasingly used for ground-based systems and lesser-criticality airborne systems (Java is a fine language and commonly used in the non-safety critical realm, however there are different opinions about Java’s runtime dynamic memory usage and safe coding standards thereof.) Conversely, when the logic is physically embedded within the silicon it is termed Complex Electronic Hardware (CEH) and when airborne, is certified via DO-254. CEH is typically written in a Hardware Design Language (HDL) such as Verilog or VHDL.

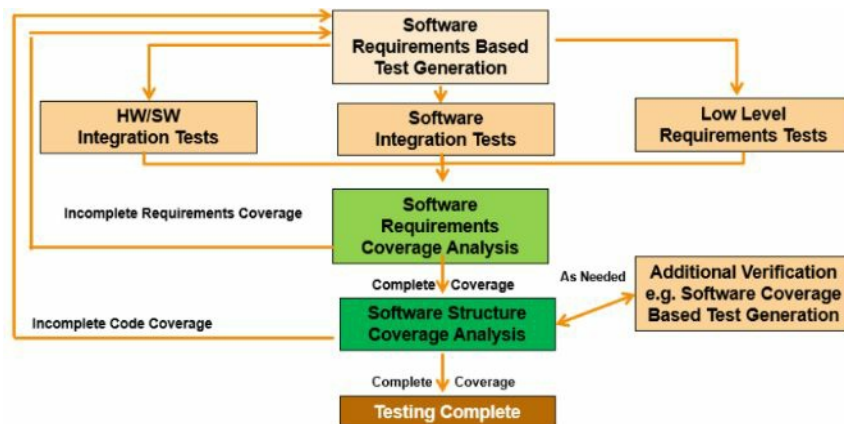
While attempts are increasingly made to verify the correctness of logic by evaluating the logic semantics, aviation takes a far more conservative and pragmatic view that such is currently not feasible for today’s complex systems. Aviation requires an ordered and well-planned approach to logic design then verification. The figure below summarizes this author’s opinion of the optimal engineering route:



It is this sequential phased approach to software development, where each major activity is depicted in a green box above, which enables the inputs/outputs to be assessed throughout the lifecycle. While V&V is shown

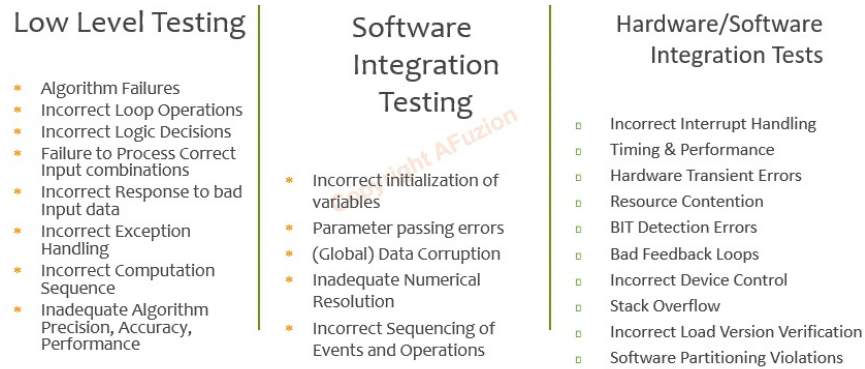
for simplicity as a single green box, the truth is that the Review portion of V&V is performed continuously and upon the outputs of each of the other phases.

The figure below depicts the logic testing and analysis process for aviation logic. Note the term “software” is used in this figure, and software can mean either software or hardware logic. Again, the term “code” (short for source code) is generally applied to software whereas the term “logic” is applied to silicon-based Complex Electronic Hardware (CEH) such as VHDL. But the testing process is similar. Avionics (airborne) hardware testers should be aware that the DO-254 document did not really address such hardware logic because DO-254 was initiated in the later 1990’s when such logic wasn’t prevalent in avionics. As noted in Chapter 8 of this book, DO-254 evolved via Advisory Circular (AC) 20-152, then CAST-27, and then the revised worldwide A(M)C 20-152A released in summer 2020. The net result of these DO-254 evolutions is essentially to treat hardware logic as if it were software code for certification purposes, beginning for DAL B.



Aviation Logic Testing & Analysis Process Implementation

The error-detection priorities depend upon the level of testing. Using the above test allocation figure, the following graphic prioritizes the test focus per level:



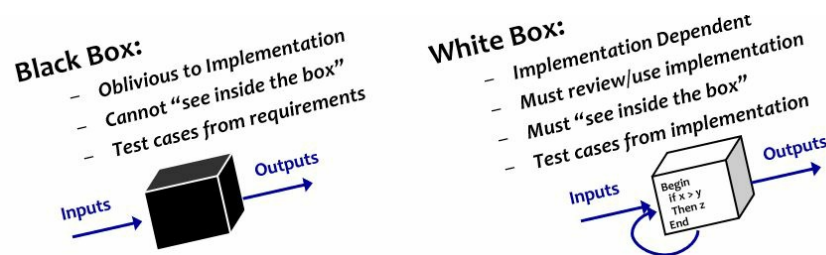
As shown in the preceding two figures, the logic testing process can be essentially broken down to the following steps while noting the feedbacks depicted via the arrows above:

1. Start with functional (requirements-based) tests by allocating individual requirements to a form of testing:
 - a. Hardware/Software Integration Tests, and/or:
 - b. Software (or Hardware for Logic) Only Tests, and/or:
 - c. Low-Level Requirements Tests (using a subset of just the software, or just the hardware in the case of logic)
2. Execute those tests per Step #1 above and assess completeness of requirements coverage including coverage of both Normal and Robustness test cases described earlier in this Chapter; if incomplete, go back and add requirement detail then repeat Step #1.
3. If DAL C or higher for software, or DAL B or higher for CEH, perform Structural Coverage Analysis as required for that DAL. If uncovered structures are detected, remove the Dead code, justify the Deactivated code (including defensive and maintenance-mode only code), and add requirements for any other code by going back and adding requirement detail to cover such uncovered structures then repeat Step #1.

Black Box versus White Box Testing.

The rigor of logic verification is commensurate to that logic's potential risk, as defined by the Item Development Assurance Level (IDAL) associated with that logic. Obviously DAL A logic is more rigorously assessed, for example, than DAL C logic. Since DAL D logic is deemed "black box" for the purpose

of reviews and tests, a review of the logic (hardware or software) is not required for DAL D. Afterall, a DAL D software failure should be readily mitigated by crew actions or other avionics, therefore such is termed “Minor” and the expense associated with logic reviews and White-Box testing of that logic is not required. However, for DAL C or greater software and DAL B or greater logic, defects can have significant operational impact therefore greater rigor is applied to both the design and verification processes. That added rigor is termed “White Box” verification meaning the actual logic is assessed directly. The figure below depicts the different between Black Box and White Box testing:



As shown above, Black Box testing means you cannot “see inside the box” and therefore testing is based upon associated requirements for the item under test. Conversely, White Box testing utilizes additional evaluations based upon the actual assessment of the source logic itself. Commonly white and black box testing are combine when evaluating logic. To prevent a tester from evaluating the logic prior to testing the requirements associated with that logic, it is best to perform Black Box testing prior to White Box testing; this enables more objective overall testing since the tester is not misled into writing test cases based upon the actual logic instead of the preferred method of writing test cases against requirements.

Aviation always requires Black Box testing and unanimous opinion is that Black Box testing is more effective than White Box-only testing, and also that Black Box testing has a greater return on investment than White Box. However, Black Box testing admittedly has the following drawbacks:

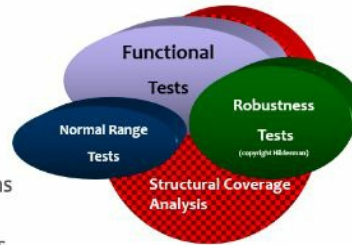
- *It is dependent upon the correctness and detail within the requirements*
- *It is dependent upon the skill of the tester to ferret out defects*
- *It is difficult to fully assess Robustness attributes inherent within the requirements (holistically) but which are incorrectly or insufficiently*

handled within the source logic.

Aviation logic then has basically four categories of testing activities as depicted in the following figures:

Four Categories of Tests:

1. **Functional Tests**
 - All Requirements
2. **Normal Range Tests**
 - “Sunny Day” conditions
3. **Robustness Tests**
 - “Rainy Day” conditions
4. **Structural Coverage Analysis**
 - Cover all code



As depicted above, functional testing equates to requirements-based tests and is Black Box meaning the tests are written based upon requirements and not by considering the logic itself. Ideally these functional tests then cover all of the normal and robustness operating conditions without the necessity of examining logic to assess such. Then structural coverage analysis is performed upon the prior functional, normal range, and robustness tests to ensure the minimum logic coverages were achieved based upon the associated DAL. And this of course happens in the perfect world, but not our real world ...

In the real world (your world), requirements are never perfectly detailed and logic developers make certain assumptions while designing and implementing their logic. For these reasons, merely performing functional (requirements-based) tests would not fully assess operational conditions that the logic was designed (or should have been designed) to handle. Therefore, additional White Box testing is necessary which assesses the logic directly after performing Black Box tests. For example, additional normal range and robustness tests may be required after the tester then examines the logic to determine if the prior functional tests were sufficient. If that process also determines that additional requirements should be added to better justify such additional test cases, then those requirements are added and requirements-based tests are then likewise added for closure. The only way to objectively determine if the testing was minimally complete is to then perform structural coverage analysis.

Robustness Testing.

Robustness testing is a somewhat (unfortunately) subjective area within aviation system and logic testing and answers the question “Is the item robust with respect to handling off-nominal conditions?” Robustness testing assesses the response to:

- ***Abnormal operating conditions***
- ***Boundary values***
 - ***Which boundaries? At and beyond bounds!***
 - ***Bit-level accuracy on boundaries?***
- ***Error & Invalid Values/Transitions***
- ***Stress & Performance Testing***

The reason robustness testing is somewhat subjective lies in the fact that the number of robustness conditions and combinations is typically larger than the available time and budget to test, particularly for complex systems where that number is infinite. For example, consider the testing of an aircraft’s DC voltage whose nominal value is 28 volts and the valid voltage range is 26.0 to 30.0 Volts. In such a case, there would be minimum five test cases:

<u>Test Case</u>	<u>Test Input Voltage</u>	<u>Test Result</u>
<i>1. Under Voltage</i>	<i>25.99 Volts DC</i>	<i>Under-Voltage Error</i>
<i>2. In-Range Low Boundary Voltage</i>	<i>26.0 Volts DC</i>	<i>In Range, Boundary</i>
<i>3. In-Range Nominal Voltage</i>	<i>26.0 Volts DC</i>	<i>In Range, Nominal</i>
<i>4. In-Range High Boundary Voltage</i>	<i>26.0 Volts DC</i>	<i>In Range, Boundary</i>
<i>5. Over Voltage</i>	<i>30.01 Volts DC</i>	<i>Over-Voltage Error</i>

Of course the above example is easy. Robustness testing becomes more difficult as myriad state-transitions and input combinations are considered. Also, performance testing, stress testing, and Worst-Case Executing Time (WCET) testing must further define expected worst case combinations which maximally stress the logic.

Logic Structural Coverage Analysis.

Structural coverage analysis is the final activity to assess test completeness against the logic. As previously noted, logic with higher criticality requires White Box testing, with the degree of White Box testing increasing along with the criticality level. For airborne software at DAL C and above, ground/satellite software at AL 3 and above, and airborne hardware at DAL B and above, White Box testing is required including Structural Coverage Analysis.

- Software: DAL C & AL 3: **Statement Coverage**

Statement coverage assesses the dynamic execution during testing of each associated source code statement, where a statement is typically the smallest entity wholly parsed by a compiler, typically denoted by semicolon (“ ; “) delineators.

- Software & Airborne Hardware: DAL B & AL 2: **Statement & Decision-Condition Coverage**

Condition coverage: Every condition in a decision in the program has taken all possible outcomes at least once.

Decision coverage: Every point of entry and exit in the program has been invoked at least once, and every decision in the program has taken all possible outcomes at least once.

- Software & Airborne Hardware DAL A and AL1: **Statement, Decision-Condition, and Modified Condition Decision Coverage (MC/DC)**

Modified Condition Decision Coverage: Every **Decision** has taken all possible outcomes at least once, and every **Condition** in a decision is shown to **independently** affect that decision’s outcome, where a decision independently affects a decision’s outcome when that condition alone affects the outcome.

The reasons for structural coverage analysis are threefold:

1. **Reason #1:** Ensure testing was performed to cover code in proportion to the DAL (criticality); and
2. **Reason #2:** Ensure no dead code exists and that deactivated code is identified and then properly handled and justified; and

3. **Reason #3:** Assess the quality of the software requirements via the functional tests of those requirements.

Ponder these three reasons above for structural coverage:

- **Reason #1** above ensures that more critical (higher DAL) logic has more of its potential paths covered by formally traced tests to each path. Therefore more critical logic has fewer possible untested operational conditions which could occur during actual operations. Remember, in complex asynchronous systems inherent in aviation, the number of input/event combinations is infinite; there is simply no way to fully test an infinite set of input combinations. Aviation's structural coverage assessment is arguably the best means to at least ensure the most likely, and obvious, combinations are covered. (Some advanced aviation engineers including this author believe Formal Methods as described in DO-333 and also Artificial Intelligence could someday lessen the importance of structural coverage. But not yet today.)
- **Reason #2** above fulfills certification objectives for White Box testing to ensure that all logic has a reason to be there (e.g. it has a bi-directional trace to a formal requirement).
- **Reason #3** above is deemed by many aviation logic testers and quality assurance to be the most important of the three reasons. As noted above, structural coverage analysis is the final testing activity, only performed after all the prior functional requirements-based testing is completed. If those corresponding requirements, and their functional tests, were developed in accordance with the aviation guideline/objectives, there should be very few remaining uncovered logic structures and paths.

Reason #3 above is a very important concept: Cert authorities, and sometimes even QA personnel, are not intimate with the item requirements and cannot readily assess the technical quality of those requirements and associated tests. But they ultimately need a mechanism to do so. That mechanism is Reason #3: if the logic requirements, and functional tests of those requirements, are both of good detail and thoroughness respectively, structural coverage analysis should readily show that 90-95% of logic structures and paths were covered via functional tests. Voila. Conversely, if 75% or less of the logic structures

and paths were covered by functional tests, it is clear that both the requirement detail and test thoroughness are deficient; go back and add requirements to achieve a 95% coverage then repeat until 100% are covered or justified as deactivated code.

Now consider the following structural coverage related definitions:

Definitions:

- **Condition:** a logical expression which is indivisible (atomic). Also commonly called a Boolean variable, which can only be equal to “True” or “False”, but cannot be divided in other simpler sub-components.
- **Decision:** a logical expression which can be composed of multiple conditions separated by logical operators like “OR”, “AND”, etc.
- **Modified Condition Decision Coverage:** every decision has taken all possible outcomes at least once, and every condition in a decision is shown to independently affect that decision’s outcome. (A condition independently affects a decision’s outcome if that condition alone affects the outcome.)

For a structural coverage example, consider the following code snippet:

```
if ( (A || B) && C )
{
/* <Insert code instructions here> */
}
else
{
/* <insert code instructions here> */
}
```

In this example, A, B and C represent Boolean expressions. In order to ensure Condition coverage criteria for this example, A, B and C should be evaluated at least one time “True” and one time “False” which is satisfied by the following two test cases:

1. A = True / B = True / C = True
2. A = False / B = False / C = False

For Decision coverage, the expression ((A or B) and C) should also be evaluated at least one time to “True” and also another time to “False”. As previously noted:

1. A = True / B = True / C = True True
2. A = False / B = False / C = False False

The above achieves Decision coverage.

Now note that Modified condition/decision coverage which requires that each Boolean condition should be evaluated once to “True” and once to “False”, while affecting the decision’s outcome. That means from one test case to another, changing the value of only one condition will change the decision's outcome. But with just the previous two test cases, it cannot be known which of the condition’s influence that decision's evaluation. Hence the need for MC/DC (see definition above). Therefore, for a decision with “N” conditions, there are a minimum of N+1 test cases required to ensure MC/DC coverage. Since the example above had three Boolean conditions (A, B, and C), the following test cases suffice:

1. A = False / B = False / C = True decision is evaluated to “False”
2. A = False / B = True / C = True decision is evaluated to “True”
3. A = False / B = True / C = False decision is evaluated to “False”
4. A = True / B = False / C = True decision is evaluated to “True”

As shown above for MC/DC:

- between the first and fourth test cases, only A changed its value, which also made the decision's outcome change its value (“False” in the first case, “True” in the second);
- in the same way, between the first and second test cases, only B changed its value, which also made the decision's outcome change its value (changing from “False” to “True”);
- between the second and third test cases, only C changed its value, and the decision's outcome also changed (from “True” to “False”).

Deactivated Code versus Dead Code.

Whereas hardware can be deemed “simple” or “complex” (see Chapter 8 in

this book for details), software is always considered inherently complex. Essentially, there are five categories of software source code (termed ‘code’ for simplicity), summarized below:

5 Categories of Code:

1. C1- Executing source code

- Normal Code

2. C2- Non-executing source code

- Deactivated Code as per DO-178

3. C3- Compile time excluded non-executable source code

- #IFDEF, smart linker, ...

4. C4- Provisional non-executable source code

- Unused libraries, RTOS functions, ...

5. C5- Design error induced non-executable source code

- Dead Code as per DO-178

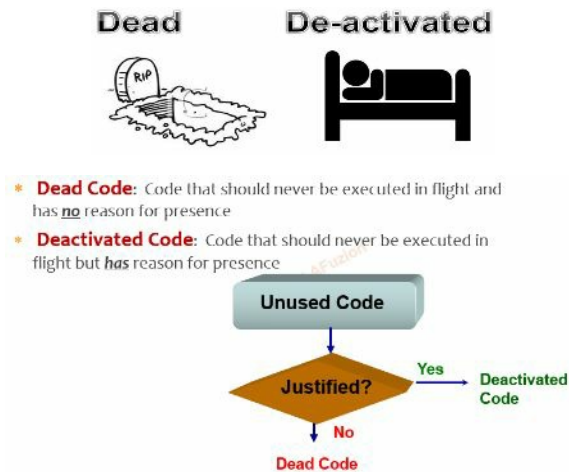
All operational aviation software is considered to belong to one of the above categories. Categories #2 and #5, deactivated code and dead code respectively, have special rules which must be applied per DO-178C and DO-278A. First, helpful definitions:

Dead Code: *Executable object code (or data) which, as a result of a design error cannot be executed (code) or used (data) in a operational configuration of the target computer environment and is not traceable to a system or software requirement.*

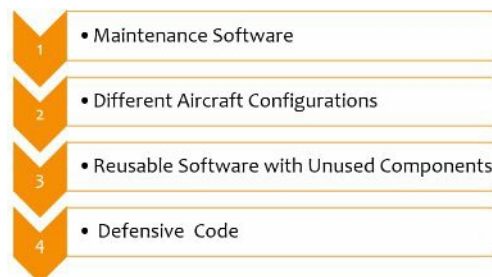
Deactivated Code: *Executable object code (or data) which by design is either (a) not intended to be executed (code) or used (data), for example, a part of a previously developed software component, or (b) is only executed (code) or used (data) in certain configurations of the target computer environment, for example, code that is enabled by a hardware pin selection or software programmed options.*

For DAL C software per DO-178C (or AL 3 software per DO-278A) dead and deactivated code need to be dealt with. Handling dead code is easy: if it doesn’t trace to a software requirement and testing and/or analysis shows it cannot be executed, it must be removed – no exceptions. However, deactivated code is essentially ‘sleeping’ code and requires additional design

and verification work. First a simple graphic to help remember the difference:

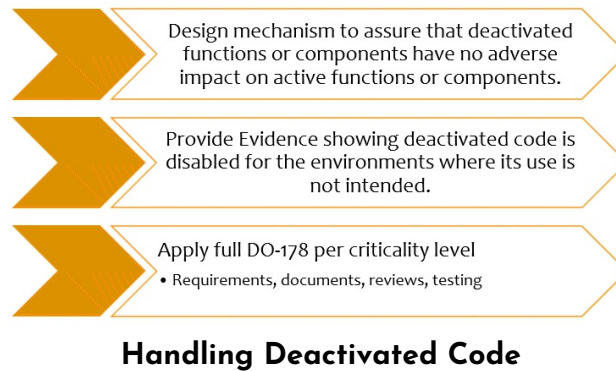


It is common, and actually desirable in many cases, for deactivated code to be present. The following graphic depicts common (and good) top four reasons for having deactivated code:



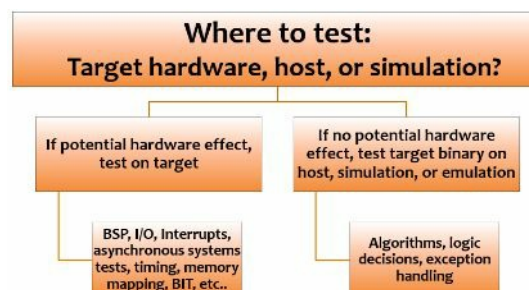
Clearly, deactivated code is desirable in the above four cases. In fact most readers should be realizing that at this point in the book, aviation software engineering is quite expensive because of the DO-178C and DO-278A rigor; Chapter 23 of this book provides more cost information. However, at the same time readers should realize that most aviation software projects make heavy reuse of previously existing software or prior configurations. If that software was well designed to be reusable with unchanged modules, the vast majority of the required activities need not be repeated on those unchanged modules with the exception of testing to show no unintended side-effects were introduced by the new configuraion or the other software which was changed. Therefore reusable software components neccessitate the usage of intentionally deactivated code. Deactivated code then must have reviewed requirements, traceability, and corresponding code analysis to show that it is truly deactivated such that inadvertent activation will not pose a safety issue. Remember: all

experienced ‘C programmers have made pointer errors and such errors could inadvertently cause the activation of any code, even code which is designated “deactivated”. The figure below summarize deactivated code management:



Where to Test

In the perfect world, all aviation testing would be conducted upon actual operational hardware in an identical finished configuration to that which is flying, or operating on the ground. And in a perfect world, we would not need seatbelts or co-pilots. In the real world, budgets and schedules don’t always allow for waiting until end of the project to perform formal testing. And logic has many robustness conditions which are hard to reach at the system level. Often testing of aviation logic is performed at a lower level than the system test such as via host development environments or simulators/emulators. However, when not using system level end-time tests, it must be shown that the logic would behave equivalently to its performance on the operational system. The following diagram helps to illustrate when each can be applied:



Hardware DO-254 Verification:

Ground-based and satellite CNS/ATM (see Chapter 11 of this Book) systems test their hardware integrated within their systems; there is no separate testing

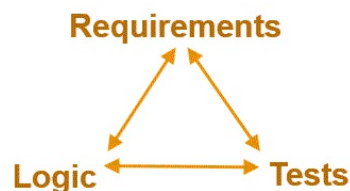
distinction for “System” versus “Hardware”. However, while airborne hardware is also subjected to integrated system testing, that hardware itself has additional verification activities for the hardware per DO-254. Those additional DO-254 verification processes are summarized in the figure below:

Tests	Reviews	Analyses
• Hardware Requirements Based Testing • Logic Testing (A/B)	• Requirements • Conceptual & Detailed Design • Logic • Requirements Based Test Results • Low Level Test Results Review (A/B)	• Traceability Analysis (A/B/C) • Requirements Coverage Analysis (A/B/C) • Low Level Test Coverage Analysis (A/B) • Structural Coverage (A/B) • General System Analysis (A/B/C) • Static Timing • Device Usage • Thermal/Power • Pin Coverage Analysis (A/B/C)

Additional DO-254 Hardware Verification Activities

It is necessary to verify airborne hardware’s timing performance within the design. This should account for the temperature and power supply variations applied to the device and also the semiconductor device fabrication process variations as characterized by the semiconductor device manufacturer. Static timing analysis (STA) with the necessary timing constraints and conditions is one of the possible means of compliance with this objective for the digital part of custom devices.

As explained above, aviation system, software, and hardware testing is multi-faceted and for complex systems, never provably complete; the number of potential test cases is infinite as you can always add one more test case. Over time, it’s common to spend more money and time on testing than was required by the original development. Therefore test automation and efficient allocation of test methods is paramount. Test coverage is provably required, along with bi-directional traceability between requirements, implementation, and tests as depicted here:



In summary, aviation testing is actually quite different from testing within Information Technology (IT), consumer devices, and other safety-critical domains. Dozens of great texts devoted solely to testing have been written; this brief short chapter should provide the tools to understand what aviation requires, and how aviation system testing differs from those other domains.

DO-160 Environmental Testing

By Vance Hilderman

Reviewed by:

- Mr. Bill De Leeuw, FAA Designee, Systems and Equipment, Software
- Mr. Alan Houp, Independent DO-160 Expert

DO-160, “*Environmental Conditions and Test Procedures for Airborne Equipment*”, applies to virtually all commercial avionics systems and many other forms of airborne equipment. In the case of DO-160, the title is quite revealing, as DO-160:

- Pertains to environmental testing, not logic execution or developmental processes
- Provides explicit, independent test criteria which must be attained to achieve equipment certification
- Applies to airborne equipment and expected worst-case environmental conditions which could potentially be encountered during aircraft operations

Essentially, DO-160 mandates tests which prove the equipment will continue to operate as desired in worst-case environmental conditions which could potentially occur in an aircraft. The purpose of which is safety, whereas commercial aspects are not important to DO-160 per se. Some people fondly call DO-160 the “Shake And Bake” test regimen, because early DO-160 testing was based upon subjecting the hardware to extreme vibration and temperature conditions. But DO-160 has always been more than “shake and bake” and the most recent versions introduce many additional forms of testing including pressure, salt, water, RF, magnetism, lightning, and many more environmental conditions.

For our European colleagues, DO-160 is very often cited as required test methods. In Europe, the ABD0100 document (Airbus) often served as the

applicable 'design standard'; the ABD0100 makes numerous references to DO-160. It is noteworthy that the ABD0100 does additionally contain design requirements/guidelines, whereas DO-160 is limited to test methods and levels. These design requirements/guidelines are, in this author's opinion, very helpful and generally necessary; however they are also considered somewhat subjective thus not typically applied by certification authorities in the USA. (Note: European Organization for Civil Aviation Equipment's (EUROCAE) ED-14 is technically equivalent to DO-160.)

Before proceeding further, please ponder a little quiz ...

1. *T / F: DO-160 applies solely to electronic hardware.*
2. *T / F: DO-160 can be used to measure service life and MBTF.*
3. *T / F: DO-160 testing is typically performed simultaneously to performance and functionality testing of the hardware/software logic*
4. *T / F: DO-160 is predominantly concerned with temperature and vibration testing.*
5. *T / F: DO-160 is a static document and rarely updated.*
6. *T / F: DO-160 testing should all be performed on the same piece of equipment.*

Do you know all the answers? A couple of them are tricky but reading the following will help.

DO-160 has a long pedigree. While the first version was released in 1975, it is derived from DO-138 which dates back to 1958, making it one of the older aviation certification documents still applicable today, albeit in its latest revision. Although many of the other "DO" documents pertain to specific aspects of hardware, software, systems, and processes, DO-160 is often considered the grandfather since almost all these systems must ultimately pass DO-160 testing.

DO-160 is essentially equipment environmental testing to Minimal Operational Performance Standards (MOPS), where testing is to be performed in a certified laboratory environment with certified & calibrated equipment. Such a laboratory environment means the tests are objective,

standardized worldwide, and repeatable. Typically these tests are performed at testing centers which are independent of the design, though larger companies may have their own dedicated DO-160 test environments. Non-certified laboratories or equipment are useful 'engineering tools' to increase design confidence and decrease actual certified laboratory test time. The successful conclusion of a DO-160 test campaign is an accepted Test Report (desirably, but not necessarily with all 'pass'). A well-written Test Report is not a trivial task, and in this context would include the certified laboratory(ies) and a list of the test articles, which includes the calibrated test equipment. The DO-160 testing would follow a previously written and customer accepted Test Procedure, being referenced in a Test Plan.

As appropriate facilities and equipment are often limited, scheduled far in advance (and expensive) resource, the savvy vendor and customer consider test facility scheduling indicative of vendor competence.

DO-160 isn't meant to be used in calculating mean-time-between-failure (MTBF) or to calculate service life (whereas Mil-Std-217, and/or an acceptable commercial variant is). Instead, DO-160 is meant to provide the minimum set of comprehensive environmental tests which onboard aviation hardware must pass, in addition to all the other testing called for pertaining to development and installation. While DO-160 cites a wide variety of MOPS, many equipment items do not require all the various tests cited within DO-160 as they simply do not apply. As with all airborne equipment the exact set of tests should be coordinated via your applicable certification authority. Tests that are 'not required' or 'done by analysis' are duly noted in the documents, then agreed to and accepted by the customer.

Of all the RTCA documents, DO-160 is ostensibly the one which is updated with the most regularity and frequency. Accident analysis, manufacturing inspections, hardware technology, and test methodologies are all continually advancing with a corresponding contribution to the body of knowledge relevant to building better hardware. For DO-160 Sections 15 to 23 (EMI/EMC) in particular, the rapidly changing specification and test requirements mandate the greatest of diligence. These sections require the "Supplier Equipment Specification" (SES or similar document), which always out-prioritizes the generic DO-160 criteria, to be scrutinized. Today,

even non-safety critical consumer products are regularly expected to be able to withstand robust day-to-day environments. Consider a typical modern cell phone: it is regularly dropped, left on hot automobile dashboards, subjected to coffee spills, and maybe even a few dog bites. Yet they are expected to keep functioning and generally do. But onboard avionics hardware has a vastly harsher environment even with the low probability of experiencing dog bites. DO-160 is regularly updated to reflect the most recent understanding of potential aviation and hardware failures in conjunction with the ever-evolving hardware manufacturing and test landscape. Ironically enough the cell phone (3G,4G, Bluetooth, WiFi, etc.) is an excellent example as to the cause of the rapidly changing and escalating requirements.

RTCA/DO-160 has been revised to address the emerging and improved test techniques. The categories that have been developed over time reflect a reasonably mature understanding as to the severity of the environmental stresses, the degrees of mitigation achievable in the design of an installation, and the robustness that must be designed into equipment in order to perform in the operating environment.

DO-160 provides criteria for the minimal standard environmental test conditions by separate environment categories. The objective of these tests is to assess, in a controlled environment, whether or not the airborne hardware is capable of functioning under contrived probabilistic environment conditions that apply for equipment onboard the aircraft per the accumulated DO-160 heritage and wisdom. DO-160 is separate from equipment performance standards that govern the development and functionality of hardware/software logic within that system. DO-160 may be used in conjunction with external standards, and notably specific modifications to DO-160 content in applicable sections (e.g. test levels, test durations).

The test levels that are to be utilized, having evolved over many aircraft and many DO-160 iterations, are dependent upon three criteria: the type of aircraft, the physical location and the Design Assurance Level (DAL). For the same physical location, the environmental test level maybe different for different DAL.

Specific DO-160 Test Categories

Under avionics software and hardware guidelines such as DO-178 and DO-254, it is never necessary to destroy hardware to verify software or hardware logic performance. In fact, if even the potential exists to harm hardware by testing, for example an over-voltage condition, then that potential alone is sufficient to cite “analysis” as the test method so as to preserve hardware. DO-160 is different.

DO-160 calls for a plethora of testing to be performed across a wide variety of environmental categories. The following lists the basic test categories cited by DO-160:

DO-160 Section	Test Category
4	Temperature
4	Altitude
5	Temperature Variation
6	Humidity
7	Operation Shock and Crash Safety
8	Vibration
9	Explosive Atmosphere
10	Waterproofness
11	Fluids Susceptibility
12	Sand and Dust
13	Fungus Resistance
14	Salt Fog
15	Magnetic Effect
16	Power Input
17	Voltage Spike
18	Audio Frequency Conducted Susceptibility
19	Induced Signal Susceptibility
20	Radio Frequency Susceptibility (Radiated & Conducted)
21	Emission of Radio Frequency Energy
22	Lightning Induced Transient Susceptibility
23	Lightning Direct Effects
24	Icing

25	Electrostatic Discharge (ESD)
26	Fire, Flammability

DO-160's Test Categories

Clearly, at nearly 500 pages covering 26 categories of tests, DO-160 is a detailed standard. If you are involved with airborne equipment development you should procure your own formal copy of DO-160, in its latest revision for general education. Due to the lengthy development of aircraft, “grandfathered specification,” be very aware that older revisions in entirety or in part may be required. Recently, RTCA removed the user guide sections and made a separate document, RTCA/DO-357 “User Guide Supplement to DO-160G”. This document aims to give rationale for requirements, guidance in applying the requirements, commentary, possible trouble shooting techniques, and lessons learned from laboratory experience.

But should your company personally perform the DO-160 testing? If you’re a large company with in-house DO-160 resources already in place, yes, by all means keep those folks busy and current. Otherwise, you’d be well-advised to outsource your DO-160 to any of the dozens of formally approved facilities worldwide which have a better understanding of DO-160 testing than you; but you still want to be smart on DO-160 to ensure you adhere to the criteria. As with all aviation product development, it’s ten times more expensive to fix a problem than prevent a problem; a thorough understanding of DO-160 will best enable you to prevent problems.

During the specification development phase, if occurring and involved in, awareness of test specification criteria, DAL, test method selection and test pass/fail criteria can make a crucial difference in product development cost. As shown by DO-160 section 22 (Lightning Induced Transient Susceptibility), a test requirement of “No upset permitted” as compared to “Upset with self-recovery” can make a substantial difference in design complexity. Once again, even if a consultant is the only option for a “wisdom-based solution,” the proper short-term consultant, albeit expensive, can save far more than the cost (as well as the nightmarish “re-qualification loop”).

Sequencing of Tests

The various DO-160 test categories cited above are independent; in other words, associated tests within each category are performed generally irrespective of the other test categories. With 26 basic test categories, there would be millions of potential combinations and sequences. It's impractical to expect equipment manufacturers to subject their products to every possible combination and sequence of environment conditions. To provide an example, it is possible for icing, salt fog, and fungus to have a small effect on radio frequency susceptibility. However, DO-160 testing for radio frequency susceptibility is performed in a static lab environment without considering the combined effects of icing, salt fog, and fungus upon that susceptibility, whereas the Highly Accelerated Life Test (HALT) will have simultaneous temperature and vibration. The order of testing is not completely arbitrary; the following rules must be followed when aggregate testing is required:

DO-160 Test Sequencing

1. Fungus resistance testing must precede salt fog testing
2. Sand and dust testing must come after fungus resistance, salt fog, and humidity tests
3. Explosive atmosphere and flammability testing must be conducted last

DO-160 testing when formally done for certification credit (Qualification) must be done on units with significant configuration management in place. That configuration management applies to released schematics, parts lists, assembly drawings and Acceptance Test Procedures (ATP or similar procedures). The unit(s) selected for DO-160 qualification testing are to represent normal production practices and are specifically prevented to be selected as to have better than typical performance. The unit(s) are to have, as a minimum, the ATP conducted during production (as is normal) and also conducted at the completion of testing. Normally the ATP is conducted several times during the DO-160 testing (qualification) such that if a failure occurs that only the ATP would detect, the cause can be isolated to subsets of the tests. Passing the ATP at the conclusion of (formal) DO-160 is a requirement.

Often due to the length of time to conduct some of the DO-160 tests, difficulty in scheduling, or the somewhat messy physical results of a few tests

(e.g. salt fog), more than one unit will be used. Conversely, units for testing can be very expensive for pressing deliveries to customers, limiting unit availability. All units used must be of the same formal configuration and same part number, differing in serial number only, otherwise the non-trivial processes are undertaken to justify the identical behavior of dissimilar configurations. As a cautionary statement, savvy customers and certification agencies will view 'too many units for testing' with suspicion, and justifiably so. It is considered normal to have one unit be used for all EMI/EMC tests, one unit being used for temperature tests (both could be the same unit of course). Having different units for cold versus hot temperature/different units for emissions versus susceptibility will not be generally accepted. Although all units may be identically configured, units must differ in the actual physical part within, even if only by innocent subtle wire routing/lengths. As such it is known that normal part-to-part variations can have very different temperatures or EMI/EMC characteristics.

If changes in the design or parts utilized are considered "large," as can easily happen, a new end item part number may be, and often is, required. This might require starting the testing over again if occurred during Qualification, or re-qualification in part or entirety; qualification by similarity is an engineering task of no small merit.

The physical unit(s) used for formal qualification is to be retained for a lengthy period of time after the completion of the testing and may be considered the property of the customer. They are not to be altered and have very restricted use, if any at all, following the testing.

DO-160 Tolerances & Test Equipment

When conducting formal DO-160 testing, the 'normal, usual and customary' DO-160 procedures and equipment is to be used; certification and calibration requires considerable expertise and resources. Even in the world of DO-160 experts, this is left to experts/special services. The specifications and tolerances are complex and specific. The test equipment used in DO-160 testing must have current, valid proof of adherence to national or international standards, be identified for make, serial number, last calibration date, calibration expiration date, or calibration validity period. Particular types of calibration records are to be maintained which may involve both

internal and external companies. As a note, the word “calibration” is inclusive of measuring, verifying performance, as often the equipment used has no “calibration” mechanism; it is either “in spec” or replaced/repairs (all of this information needs to be included within the test documentation, certainly the DO-160 test report, which is often a portion of the Qualification Test Report [QTR]). For these reasons, it is common to outsource testing to any of the numerous independent test entities that perform DO-160 testing. A good QTR writer is expected to produce the required report and be accepted by the customer; although typically there are questions and revisions. An inexperienced QTR writer can cause problems similar to an inexperienced test conductor or an inexperienced Test Procedure writer. A mentoring relationship is paramount.

DO-160 Ambient conditions

For the various tests, DO-160 requires testing within what is termed “acceptable ambient conditions”. Obviously, some of the tests clearly change those conditions, for example when conducting temperature tests, icing tests, altitude tests, etc. But unless the test conditions require changes in temperature, humidity and pressure, the following conditions should be maintained for DO-160 testing:

- Relative humidity should be less than or equal to 85%.
- Temperature should be between 15-35 degrees C.
- Pressure should represent standard pressure of -1,500 feet to +5,000 feet, i.e. 84 – 107 kPa.

As an interesting note, all too many test facilities fail to have certified and calibrated devices to measure ambient conditions on display and used during the tests.

DO-160 Notes

DO-160 testing, while extensive, conducts one environment test at a time. However, the “real world” clearly needs multiple tests to be executed simultaneously for simple schedule pressure considerations. Fortunately, DO-160 was created and is maintained with this in consideration. To offer some insight into this highly desirable yet highly difficult task, consider that the

test levels/duration imposed by a single DO-160 test likely greatly exceeds the similar single element level occurring on the aircraft. The environment synergy in a real-world vehicle, numerous vehicles and a vast number of designs, is not practicably specifiable. It is important to have an understanding of this. It is also good to know that for the same environment, the limits for emitting/generating are typically much lower than susceptibility limits. Again this is due to the synergy on aircraft of multiple sources and to provide statistical (probabilistic) margin abilities. For example, the EMI-conducted emission level limits are likely much lower than the EMI-conducted susceptibility level.

In summary of levels and limits:

- The applicable DO-160 levels are based upon aircraft type, unit(s) location (one unit is connected to other units, in various locations), DAL, and apply a probabilistic understanding.
- Testing is done on a single environment at a time; the real-world combinational effects are accounted for in DO-160 by the level and duration.
- For the same test environment, the permitted emission limits are much lower than the required non-susceptibility limit; arguing to raise the permitted emission limits in order to pass the test based on this is rarely accepted. In principle this applies to vibration as well as shock.
- Many of the specifications, or test measurement metrics, are done in dB (decibel)—a 3 dB change in a power measurement is a factor of 2.

In DO-160, several of the sections contain specific directions to detect potential design weaknesses, susceptibility, and transmission of deleterious emissions. Analysis for sensitive frequencies to be tested (including mechanical resonance), and testing in multiple modes of operation for units with multiple modes of operation can be pertinent. Since some modes of operation have durations which are very short compared to test time (e.g. startup), steady-state modes are utilized. If analysis does not reveal special frequencies of interest, then default values are used.

In comparison to DO-178 or DO-254, DO-160 has fewer subjective pitfalls.

DO-160 does have, or attempts to have, clear pass/fail criteria which are to be specified in the test procedure and are to be reproducible. A test failure must be addressed and resolved. Resolution can become a messy process, but not necessarily so. Test failure resolution/correction methods include retest with pass (with justification), description of anomalies test aberrations, test equipment substitution (with justification), request for specification relaxation, design alteration, and re-qualification.

DO-160 applies to most all airborne equipment, however the type of aircraft for which the equipment is intended must be considered. For example, rotorcraft has a more extreme vibration and shock environment than does commercial fixed wing aircraft; thus corresponding DO-160 test criteria are more rigorous for helicopters. Also, there have been recent attempts to harmonize some DO-160 testing with relevant military standards such as MIL STD 810. In general, it is desirable to use or reference similar standards and the user may do so if they prove such standards meet or exceed the threshold criteria cited by DO-160.

Avionics is continually evolving and DO-160 cannot be updated in real-time to immediately accommodate all evolutions; Technical Standard Orders (TSO's) should be consulted for specific equipment types. Every few years RTCA compiles the suggested DO-160 revisions, generally related to new technology, and publishes a new revision. For example, evolutions in GPS, GNSS, and new solvents/fluids precipitated the need for a revision, just as AFDX, cellphones, and consumer electronics on board future aircraft will drive additional near-term updates. Also, ongoing crash investigations may yield ancillary analysis necessitating the need for improved testing, ostensibly via DO-160.

Quality & Process Assurance in Aviation

By Vance Hilderman

Reviewed by:

- Mr. Don Coleman, FAA DER
- Ms. Gamze Eroglu, Senior Certification Engineer
- Mr. George Meier, FAA DER

What & Why Quality/Process Assurance in Aviation.

Quality Assurance (“QA”) is arguably the most critical aspect of avionics software and hardware certification within aviation development. And whether it’s termed “Process Assurance” (PA) for aircraft, systems, and hardware, or Quality Assurance for software, the importance of QA/PA cannot be understated. However, QA/PA is rarely given the attention or credit befitting its crucial role. And in fact, aviation-related QA is quite different than QA in traditional industry and consumer product development. Consider the following statements and assess whether they are true or false; answers and explanations follow:

1. *T / F: QA’s most important role is assessing final product quality.*
2. *T / F: QA personnel perform technical reviews.*
3. *T / F: QA personnel assess avionics development engineer’s adherence to criteria specified in the required version of DO-178C, DO-254, and/or DO-278A.*
4. *T / F: QA plays a key role in review of requirements, design, and code, and also conducts tests.*
5. *T / F: The four Stage Of Involvement (“SOI”) events represent the four QA audits of the avionics development process.*
6. *T / F: Every Level A and Level B avionics development project requires at least three different persons, and the most crucial of those for certification is the independent QA person.*

If the above questions were truly easy, congratulate yourself on your genuine QA knowledge. If they simply seemed easy, then the information that follows

is for you. In fact, answering the above without understanding the overall quality assurance role in aviation development is like understanding Fourier transforms without first understanding The Calculus: impossible for mere mortals ...

In aircraft, systems, hardware, and software development, QA/PA (hereafter termed simply “QA”) has three primary responsibilities:

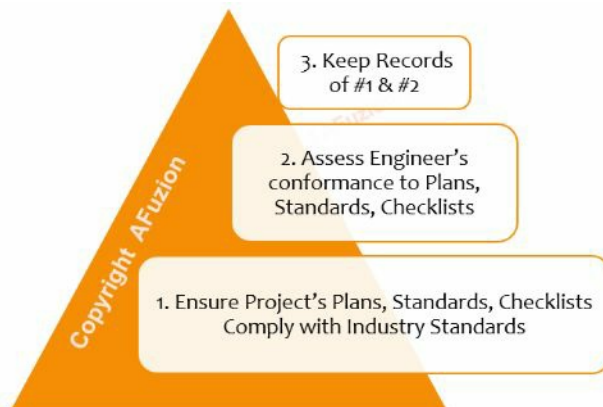
1. *Ensuring the project specific engineering plans and standards comply with DO-178C/254/278A, and ARP4754A.*
2. *Assessing, and then ensuring that the independent engineering organization has followed those plans, from project inception through completed delivery.*
3. *Build and retain evidence of the above via recorded Audits with resolution of all issues, defects, and process improvements encountered during*

In other non-avionics development activities “Quality Assurance” often implies a more adjunct, and more passive, measurement role. Wikipedia, for example, aptly states QA “*is the systematic measurement, comparison with a standard, monitoring of processes and an associated feedback loop that confers error prevention.*” In most industries, QA reports to Engineering, assists with documentation technical reviews, and executes various system/software tests near the end of the development lifecycle; however none of these are the roles of QA within aviation. Why doesn’t aviation QA perform these traditional industrial and commercial consumer industry QA roles? The weaknesses of such a traditional, yet common, QA interpretation for other industries if applied to aviation:

- Who ensures project plans and standards are correct and compliant to ARP4754A, DO-178C, DO-254, DO-278A, etc.?
- Who ensures applicable processes are deterministic, repeatable, clearly defined, and compliant to ARP4754A and DO-XXX?
- Who ensures the engineering process feedback loop has been defined correctly and then followed?
- Who is responsible for ensuring proof exists that errors were

appropriately identified, dispositioned, and resolved3w?

In aviation software development (e.g. DO-178C and DO-278A, see preceding chapters), the answer to the above is simple: “Quality Assurance.” QA is based upon what this author terms the “QA Pyramid” as depicted below; note the foundational elements versus the ensuing QA tasks on top of this QA foundation:



Software Quality Assurance Primary Responsibility Pyramid

However, at the aircraft, system, and hardware level, quality assurance is instead termed “Process Assurance” (PA) as additional supplier auditing and manufacturing assessments are performed; thus an equivalent PA pyramid is depicted as below:

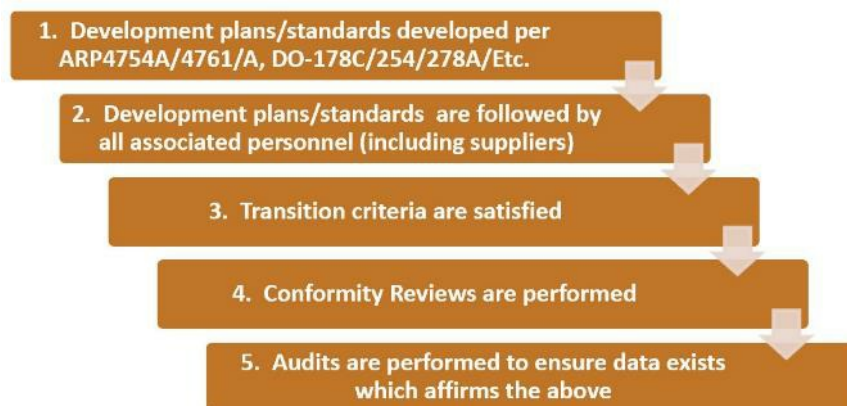


Systems & Hardware Process Assurance Primary Responsibility Pyramid

ARP4754A and DO-XXX are somewhat “flexible” regarding the manner in which QA processes are defined, scheduled, and performed. However, while flexible, aviation requires defined and measured processes be methodically applied to ensure product quality. Not merely to improve product quality:

improving a weak product until it became a mediocre consumer product is clearly insufficient within aviation. Clearly, the quality of aviation development depends upon the quality of components comprising development. Which components of the aviation development process affect quality? All of them. Are all those components equally important? No and yes: “no” they do not all have the same potential impact upon quality, but “yes” since each component CAN affect quality then each component has equivalent status as a potential failure point within development. Avionics quality assurance then is tasked with applying the industry guidelines while reviewing all aspects of the avionics development process to determine how the inputs and outputs to that process should be independently assessed. The avionics development ecosystem starts with Safety and Systems, and then the Hardware and Software are considered as subsystems. Therefore, QA must take both a big-picture view of that entire ecosystem, while also addressing the smallest engineering processes which can affect product quality. It starts with quality assurance planning ...

The Quality Assurance Plan (“Process Assurance Plan” for DO-254 and ARP4754A), typically authored but always signed by QA, is one of five project-specific DO-XXX plans (eight plans for ARP4754A; see prior respective Chapters in this book) which define how that project intends to meet the applicable DO-XXX objectives. But Quality Assurance needs to be in place early, during the Safety and System definition phases, to ensure those phases are compliant to applicable processes with proper artifacts in place for subsequent software and hardware development. While flexible in process, aviation guidelines are not pushovers; QA must ensure the following objectives are met:



Key Process/Quality Assurance Activities

Upon first glance, the above list of aviation QA objectives seems easy; almost obvious. In fact, in other, non-aviation development environments, “quality assurance” appears to embody similar objectives: assess product implementation to measure and improve quality. As a result, virtually every consumer electronics product manufactured today has some form of basic quality assurance performed upon it.

However, closer examination followed by understanding of aviation’s QA framework reveals more proactive and robust QA guidance. In avionics, there are five required plans for every safety-related airborne software system (and a corresponding five plans for custom silicon-based complex hardware). While all five plans are important (and FAA/EASA wisely decline to state “which” plan or objectives are most important), avionics certification experts generally agree that the Certification Plan is the most important plan, with the order of importance of the other four required plans as follows:

1. ***Certification Plan***
2. ***Quality Assurance Plan***
3. ***Configuration Management Plan***
4. ***Verification (and Validation for hardware) Plan***
5. ***Development Plan***

As shown in the preceding list, the 2nd most important plan is the QA Plan which must describe QA processes which ensure each system:

- has a complete set of plans/standards which embody 100% of all applicable DO-XXX objectives;
- defines processes to monitor development according to the Plans and associated Standards, including “transition criteria” which define entry/exit attributes for each engineering activity;
- has mechanisms to assess whether the actual processes used throughout the development process complies with those plans;
- has a feedback and control process to ensure corrective measures

accompany relevant defects, with record keeping to prove status and conformity.

In aviation, each of the five key areas of aircraft and systems development has their own QA plan as follows:

- **Aircraft** “*Aircraft Process Assurance Plan*”
- **Avionics Systems** “*System Process Assurance Plan*”
- **Avionics and CNS/ATM Software** “*Software Quality Assurance Plan*”
- **Avionics Hardware** “*Hardware Process Assurance Plan*”

While different documents with differing assurance scope and evidence, the above plans are actually quite similar and address the key topics depicted in the following figure:

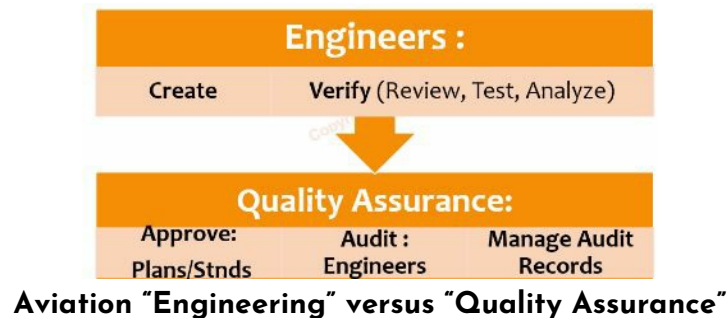


Key Quality Assurance Plan Topics

The Quality Assurance Role

Many companies, and certainly traditional companies with a track record of reliable development, are adept at defining a quality system and following it. However, DO-XXX places the emphasis, or burden of proof, upon the Quality Assurance department. The reason for this is both technical and politically pragmatic: where multiple people and departments bear equal responsibility for proof of quality, finger-pointing and a lack of true accountability may ensue. Where instead one department is responsible, the authority is clearly designated: independent Quality Assurance. Whereas QA in non-aviation industries participates in technical reviews, executes tests, and reports to engineering management, aviation QA is decidedly different;

the following figure depicts the explicit differences between Engineering and QA within aviation:



So, does QA bear sole responsibility for quality such that all the project engineering staff can relax and choose not to follow the plans and standards? Of course not; engineering must follow all applicable plans and standards as assessed by technical reviews and quality assurance audits. Does QA bear the responsibility to assess technical compliance? No; technical compliance is assessed via Reviews, and Reviews are done by engineering. QA does not perform technical reviews; instead, QA performs audits to assess engineering's adherence to the defined process. Wait, isn't this really verbal nitpicking by thus differentiating between Reviews and Audits? Yes, QA "reviews" engineering process adherence by performing audits. But this is not called "Review" since in DO-XXX Reviews are thorough and technical in nature and thus performed by Engineering. Remember, engineering performs Verification which includes Reviews, Tests, and Analysis. Therefore, engineering is responsible for performing the reviews of engineering artifacts including requirements, design, code, and tests. QA performs audits to ensure engineering follows the review processes contained in the Verification Plan.

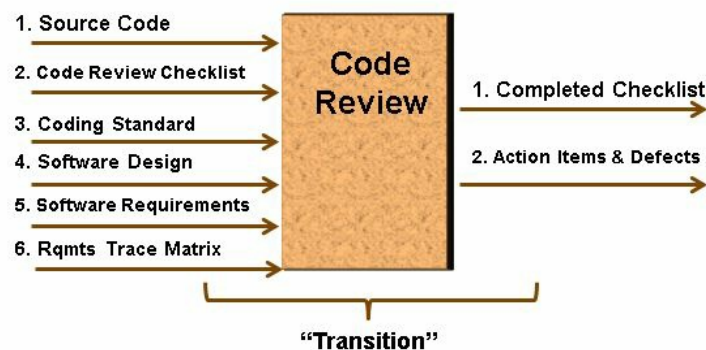
Quality Assurance Audits

QA's audits are not performed on each requirement, design, code, and test as are engineering reviews. This is an all-too-common area of difference between DO-XXX and other standards which have QA taking an active (some would say "leading") role in technical reviews. By definition, an engineering review is fully applied against a pre-defined checklist for all requirements and tests (and design/code for higher Design Assurance Levels (DAL)). On the other hand, QA audits are sampling of each engineering

process which primarily includes auditing the following four engineering activities:

1. Development processes;
2. Configuration Management processes;
3. Verification processes;
4. Engineering reviews associated with the Development and Verification processes

So, QA audits are process-oriented whereas engineering reviews are technically oriented. Reviews (done by Engineering) and Audits (done by independent QA) are performed against a pre-defined checklist template, with records kept of the corresponding review and audit. QA also must audit transition criteria for each engineering activity, meaning the Entry and Exit criteria as depicted in the following diagram:



Aviation Software Transition Example: Code Review, as Audited by QA

The QA audit records (checklists) must be kept in a formal Configuration Management system according to the associated project Configuration Management Plan. The Quality Assurance process, including the auditing process, is performed according to the guidance provided in that Quality Assurance Plan.

QA Audit Sampling & Sample Size

QA Audits are “audits”, not “re-reviews”. QA audits must include at a minimum all of the following lifecycle phases and artifacts:

- *Aircraft, Safety, Systems, Software, and Hardware Development*

including safety assessments, requirements, design, implementation, integration, configuration management, and V&V.

- *Plans, if not written by QA, must be fully reviewed and approved by QA*
- *All Aircraft, Systems, Software, and Hardware requirements, traceability, design/code/tests, ancillary files, verification records, bi-directional traceability, and processes including transition criteria*
- *Test plans, procedures, test cases, and results*
- *All third party tool documentation*
- *Tool qualification plans*
- *CM Records*
- *Peer reviews*

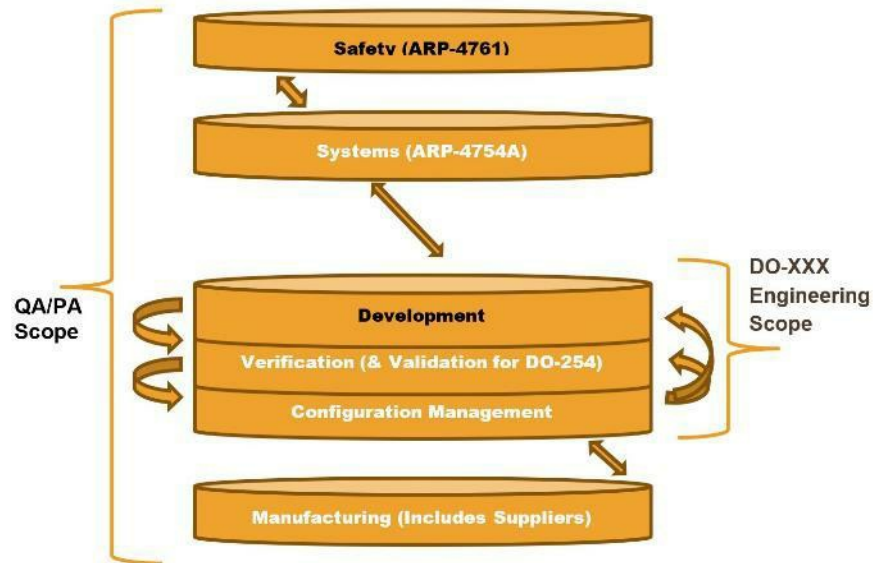
While a small number of companies devote ample resources to Quality Assurance and strive for 100% auditing, most companies simply lack such resources and are constrained to audit a subset. A good rule of thumb is “5-10%”, meaning 10% of the engineering artifacts are audited by QA. However, it is best to keep in mind the following:

- Audit sample size should be greater than 5%;
- Audits should sample at least one artifact per engineer per phase; for example, if there are three engineers; designing hardware, then each of the three engineers will have at least one of their designs audited;
- Audit sampling size and frequency should increase when QA findings increase;
- Audit sample size and frequency should be pre-defined within the QA planning documentation;
- Audit findings must be recorded and tracked through the update process; if the audit finding should result in a process update (in addition to the artifact update), such needs to be additionally managed and tracked through completion by QA.

- For small projects, audit sample size should increase to 20%+.

If a company already has a strong culture of corporate quality assurance, do they obtain extra credit for such within avionics development? The answer is a clear “No”: in aviation, each product version is typically treated as a standalone entity and each product’s quality is assessed based upon the attributes of its associated developmental aspects, not more ephemeral corporate aspects which may be unrelated. For example, just because a company has world-class near-perfect quality assurance capabilities does NOT mean those capabilities were fully applied to a given product. In fact, it’s always possible shortcuts were taken and that companies with great corporate quality assurance capabilities did not fully apply them to a given product. That is why the avionics development ecosystem is imbued with an underlying “guilty-until-proven-innocent” philosophy which translates to quality assurance. Having stated such, companies with strong cultures of corporate quality assurance should find the aviation development guidelines fairly straight-forward and easy to adopt. Especially when that corporate quality assurance cultures addresses engineer training, job management, artifact identification and retention, formal reviews, and appropriate configuration management controls with record keeping. Experience indicates that companies with strong corporate QA cultures are already applying 70%+ of the requisite DO-XXX quality assurance criteria.

On typical aviation projects, individual engineers focus upon development within their area of specialty including: safety, system requirements, architecture, software, hardware, or verification; the project manager and technical director keep the “big picture” in mind and guide the activities of the individual engineering teams. However, as Quality Assurance is independent of engineering, QA must itself be fully aware of an even larger “bigger picture”. QA must ensure all elements of Safety, Systems Engineering, Development, Verification, Configuration Management, and Manufacturing are defined, assessed, and recorded. Pictorially, these relationships are ideally depicted as shown on the following abbreviated graphic:



With increased technical complexity in aviation comes increased emphasis upon QA to ensure that complexity is appropriately managed. Whereas the DO-XXX guidelines contain good engineering process benchmarks, they are noticeably “light” on corresponding QA benchmarks. Why the vagueness for QA criteria? Remember, QA’s role is to ensure the engineering plans/standards are sufficient in defining how DO-XXX is to be implemented, then assessing engineering’s conformance to those plans/standards. DO-XXX is not a recipe book stating how processes are to be defined or followed. Other than defining and assessing the engineering inputs and outputs, QA activities are dependent upon many variables which differ widely between projects: size, scope, complexity, tools, criticality, automation, etc. It is QA’s job to consider these variables and define the appropriate measurable process criteria within the engineering plans/standards. Many recent trends increasingly address QA including Total Quality Management, Continual Improvement, Team Based Improvement, Agile Methods, etc. All of these trends have something to offer and should be considered within the normal evolution of QA. However, as they are largely subjective the barest essence of QA within aviation remains unchanged: ensure engineering processes are sufficiently defined, and then followed, resulting in provable and secure quality systems and aircraft.

**A Final Reminder On Avionics Systems and Hardware
“PA”**

ARP4754A addresses the framework of aircraft and systems development. DO-254 is principally concerned with the engineering processes surrounding the production of complex electronic hardware, more commonly known as “firmware”: silicon-based logic. Aircraft, systems, and hardware processes are similar to those of software in DO-178 for the simple reason that DO-178 was used as a reference when the other domain guidelines were being written. However, quality assurance aspects of aircraft, systems, and hardware have three fundamental differences with software, namely:

1. Aircraft, Systems and Hardware development covers a broader lifecycle range of processes, thus “quality assurance” is termed “process assurance” within ARP4754A and DO-254;
2. The role of suppliers in aircraft, systems, and hardware development has a greater contribution to product quality, thus ARP4754A / DO-254 require more thorough auditing of suppliers.
3. The role of repeatable manufacturing in hardware development quality possesses more vagaries for hardware than for software; thus ARP4754A / DO-254 requires more thorough auditing of the manufacturing processes.

For these reasons, quality assurance for hardware (as well as for aircraft and systems via ARP4754A) is termed “Process Assurance” as there is a broader range of processes to define and then audit via such Process Assurance. Audits must additionally be performed to ensure the aircraft/system/hardware item used for conformance assessment is built in compliance with the corresponding aircraft/system/hardware life-cycle data. An inspection should thus be performed to ensure the completed final aircraft/system/hardware complies with its design data; an example of this activity would be a First Article Inspection.

Military Aviation Compliance

By Vance Hilderman

Reviewed by:

- Mr. Emmett Dignan, Senior Avionics Engineer Northrop Grumman
- Mr. Vincenzo Mirra, Aerospace Quality & Certification Manager, Leonardo Company

Defense organizations throughout the world are increasing their adoption of DO-178C (Software), DO-254 (Hardware) and ARP4754A/ARP4761 (Aircraft/Systems Safety). Why, and what are the implications?

For decades, military organizations have developed hardware and software using a variety of specialized, defense-oriented standards including 2167A, 498, and 882. As Military organizations, they were highly motivated to use hardware and software standards which differed from the commercial sector since it was perceived that military applications were “different.” Militaries utmost concern was primarily “Mission”. Today however, there is an accelerating momentum toward Military/Commercial avionics convergence: *adopting DO-178 and DO-254 worldwide along with the systems and safety guidelines ARP4754A and the new ARP4761A*. Today, fighter jets (Joint Strike Fighter, T-50, etc.), cargo/refueling planes (C-130, C-17, A400M, KC-46, etc.) and UAV/UAS’s (formally called RPAS: Remotely Piloted Aircraft Systems) are requiring compliance to DO-178 and, increasingly, to DO-254 and ARP4754A/ARP4761.

What are DO-178 & DO-254?

As already covered in prior chapters of this book, DO-178C is the fourth iteration of the FAA’s avionics software standard, required for all commercial airborne software, which contributes to safety of flight by ensuring with a sufficient level of confidence that the software performs its intended functions that have been assigned by the system requirements. For over twenty five years, commercial avionics software has required

certification via DO-178, then DO-178A, then DO-178B, and now DO-178C. But in the early 2000's certification authorities realized that avionics safety was dictated by both software and hardware; hardware was just as important as software, but only required adherence to DO-160, the environmental testing standard. So SC-180, the precursor to DO-254, was initiated, thereby levying consistent certification requirements upon hardware. The basis for DO-254 was DO-178B, ensuring similarity between certification of software and hardware in terms of processes and objectives to be satisfied. At a higher level, ARP4754A applies to civil aircraft and systems, while ARP4761 (and the new ARP4761A version) provides corresponding safety analysis details.

DO-178 (software) and DO-254 (hardware) presume that software and hardware must operate in harmonic unison, each with proven reliability. Previously, hardware was considered "visible" and tested at the system level with integrated software; hence hardware was exempt from DO-178 quality attributes. But that exemption resulted in functionality being moved from software to hardware for the purpose of avoiding hardware certification. Additionally, hardware complexity has evolved such that hardware is often as complex, or more so, than software due to the embedded logic within the PLDs, ASICs and FPGAs. Now, everyone recognizes that hardware and software comprise an inextricable chain with the quality equal to that of the weakest link, thus the mandate to also apply DO-254 to avionics hardware.

DO-178, DO-254, ARP4754A, and ARP4761 utilize five different levels of criticality, ranging from Level A (most critical) to Level E (least critical); these are officially termed "Development Assurance Levels (DALs)". Each avionics system is assigned one or more levels of criticality based upon a system safety assessment which analyzes each system's potential contribution to aircraft safety; each hardware and software component within that system must meet or exceed its assigned criticality level. As the criticality level increases, so does the degree of rigor associated with documentation, design, reviews, implementation, and verification.

Military versus Civil Aircraft.

Previously, military organizations throughout the world utilized their own standards for hardware and software development. Their rationale for such was:

- *Military projects were more complex than commercial projects.*
- *Mission completion is always a highly desirable goal and surpasses “safety” in some instances.*
- *Military projects needed higher quality than civilian projects.*
- *Military projects had numerous suppliers to manage.*
- *Military projects required specialized military/sensitive functionality and complex integration cycles.*
- *Military projects had long airframe lifetimes to account for.*



Granted, prior to DO-178 in the 70's and 80's, the above rationale was valid. However by the 90's the above rationale gradually eroded. Today, consider the commonality between Military and Commercial avionics:

- *Both utilize high complexity and complex integrations.*
- *Both utilize hundreds of suppliers (many supplying nearly equivalent avionics to both Military and Commercial clients) with long project lifetimes.*
- *Both require access to leading-edge commercial technologies.*
- *Both are increasingly concerned with re-usability, quality, and increased cost-effectiveness.*
- *Both require a high level of operability, reliability, maintainability, and safety.*
- *Military aircraft are now utilized more and more in commercial airspace (they do not want to be restricted in flight paths or hours).*

By the early 2000's, U.S. military organizations realized that the commercial aerospace sector, particularly those regulated by the FAA via DO-178, maintained certain advantages, advantages not inherent in the defense establishment. They were faced with a choice:

1. Maintain the status quo and do nothing.
2. Update their own Mil standards to adopt the best aspects of DO-178.

3. Simply adopt DO-178B (and later, DO-254 and DO-178C).

What choice was made? Option #3 above. Was it a simple choice? No: as with any established organization, there were myriad opinions, entrenched practices and opposition, added initial transition costs, and politics. The result? A gradual adoption of DO-178, DO-254, ARP4754A, and ARP4761A (though DO-254, ARP4754A, and ARP4761 adoption in the Defense sector lags that of DO-178). Further complicating the military adoption of civilian aviation guidelines were the following aspects:

- *The military did not want to relinquish oversight to the Federal Aviation Administration (FAA), nor did the FAA have the bandwidth or authorization to intervene within Military projects.*
- *Military organizations were unfamiliar with DO-178 and DO-254 specifics and therefore applied widely varying and subjective criteria; truth be told, DO-178 and DO-254 are terse and vague—specialized training by experts is typically required to apply them in the real world.*
- *Because of the above, militaries often further complicate DO-178/DO-254 adoption by requiring simultaneous adherence to their own Mil standards in addition to DO-178/DO-254! While well-intentioned, this action is counter-productive since the standards differ and conflict with each other in key areas; DO-178 and DO-254 already have ample ambiguity and subjectivity which is grossly complicated when requiring corollary adherence to Mil standards.*
- *Military safety was commonly built around MIL STD 882, which is somewhat different than ARP4754A/ARP4761A.*

Sample Military Aircraft with DO-XXX Compliance.

Today, numerous military aircraft have all achieved at least partial compliance to DO-178 and DO-254. To this author's knowledge, none of them have achieved meaningful compliance with ARP4754A and ARP4761/A; not because the airframers tried to avoid ARP47XX, but rather because ARP47XX wasn't "on the radar" at the time of the corresponding airframe development. Instead several of these are aircraft did comply with MIL-STD-882E (more on that below). Examples of such military aircraft

with at least partial DO-178B/C, and some with DO-254, compliance are shown below:

- **C-130, C-17, B1, B2:**
Many new and reverse-engineering
avionics systems, per DO-178B:



- **F-35: Most avionics
systems: DO-178
Equivalent:**



- **UAV's:**
Global Hawk & Predator
Most systems "Compliant":



- **KC-146**
Civil airframe, military
equipment DO-178 Compliant:

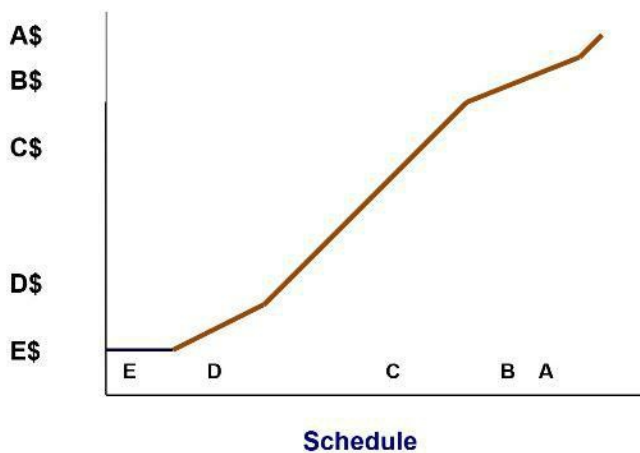


Unlike Military standards, DO-178 and DO-254 utilize five different criticality levels. Why? Cost. Purely cost. If cost were no object, all avionics software would be designated Level A, the most critical level with the strictest requirements. However, each of the dozens of avionics systems onboard aircraft does not affect aircraft safety to the same degree. The criticality level is chosen via analytical processes which assess the contribution to aircraft safety of each system, sub-system, and component; this criticality level is also based on a combination of engineering judgment, flight experience, and system service life. These safety analyses are covered by their own standards including ARP-4761 (ARP-4761A beginning in 2020) for DO-178/DO-254 and are well-known throughout civil aviation. However, such safety analyses are relatively new to Military avionics, hence the criticality level selection subjectivity within defense projects.

Regarding cost, the following graph accurately depicts the cost-delta associated with the different criticality levels:

Graph: Delta Cost and Schedule of DO-178 per
Criticality Level

Criticality Level - Cost



Costs versus schedule by criticality level are direct consequences of the number and complexity of objectives to be satisfied, however other factors are relevant; it is also a matter of what and how the applicant (industry or organization) perceives and understands the applicable “DO” guidelines as such often yields misunderstanding, mistakes, re-planning and re-work. This is a major cause of the cost increases for aviation compliance.

A popular myth is that DO-178 is expensive. As can be seen above, Level D certified software still has full planning, requirements, implementation, reviews, and basic testing processes applied. Plus configuration management, quality assurance, and DER liaison are applied to Level D. But the costs of Level D are hardly more than any non-certified commercial software process. Why? Because Level D is comprised almost entirely of normal industry-standard software engineering principals.

Another myth is that the most significant cost escalation occurs when moving from DO-178 Level B criticality to Level A. Untrue.

The cost impact of DO-178 is most significant between Level D and Level C. Why? Level C requires the following which Level D does not and which results in Level C requiring at least 30% more budget and schedule than Level D:

- Testing of low-level software requirements
- Ensuring 100% coverage of all statements
- Greater rigor placed upon reviews

- In many cases more rigorous configuration management
- Far more complete traceability is required via 178/254 than traditional Mil standards; most projects use a tool.

Level B requires additional structural coverage (decision-condition, i.e.. all branches in the source code), additional independence in reviews, and tighter configuration management. On first glance then, it would seem that Level B should be significantly more expensive, roughly 50-70%, than Level C. And in theory, such might seem to make sense. But as in many areas of life, common sense overcomes theory. In Level B (and C), there must be detailed low-level software requirements and they must be thoroughly tested. During such requirements-based testing, the great majority (70-90%) of the branches is already executed and hence requires no additional structural coverage testing if test capture and coverage tools are appropriately used! Therefore, the seemingly significant cost increase associated with Level B versus C structural coverage is already mitigated by requirements-based testing. And quality software engineering organizations already incorporate a semi-automated and streamlined process which includes independent reviews and tight configuration management; hence the added cost of those aspects for Level B is largely mollified. The reader is well-advised to undergo upfront DO-178 Training and DO-178 Process Improvement to leverage these cost reduction techniques.

Level A is the most critical software level and hence the most expensive. True. But another myth exists for Level A, namely *“Level A is extremely difficult to achieve and will cost at least 30-50% more than Level B.”* False. Level A imposes yet more structural coverage requirements (MCDC testing), robustness and correlation requirements, and slightly more stringent reviews. The most significant cost driver over Level B is the MCDC testing requirement. However, with proper application of modern structural coverage tools, personnel training, and thorough requirements based testing, the added cost for Level A can be largely contained.

The aforementioned cost deltas are actual, achievable results, as documented by this author on dozens of successful projects as well as leading aerospace providers. However, these cost results are NOT the industry average: the average DO-178 avionics project exceeds these cost deltas by 20-50%. Why?

Because of inefficiency, misunderstanding of DO-178, and not applying “Best Practices” to contain costs. This results in re-work and over-work.

Safety: ARP4761 and ARP4754A

Traditionally, militaries have applied their own safety standards, such as the ubiquitous MIL-STD-882 E. However, like mixing oil and water, the military safety standards do not mix well with civilian software/hardware guidelines. ARP4761 and 4754A are much better suited to integration with DO178C based upon their Development Assurance Level (DAL) assignment, particularly Functional DAL (FDAL) and Item DAL (IDAL) approaches. And the civilian application of the Preliminary Aircraft Safety Assessment (PASA) and Preliminary System Safety Assessment (PSSA) are quite unique and not as well covered in 882 E. (See AFuzion’s separate papers on ARP4754A and also ARP4761). When the new update to 4761 (of course: “ARP4761A” being released in 2021, worldwide militaries are expected to more fully standardize on the civilian counterparts.

Compliance versus Certification, and differences with Military Compliance

Since the FAA, with very few exceptions is not involved in Military projects, formal certification is not required. Instead, military agencies typically self-certify under the term “*Compliance*”. Thus, militaries require compliance to DO-178/DO-254, not certification. The difference? In Compliance,

- *The FAA is not involved,*
- *DERs are not required, though they are advised, and*
- *Certain requirements are relaxed, including the safety analysis, code robustness, strict MCDC proof of all instances, etc.*
- *Militaries are more likely to waive, or lessen the acceptance threshold, of certain DO-178C and DO-254 objectives including tool qualification criteria, robustness testing, low-level requirement detail, and transition criteria adherence.*

Synopsis of Other Standards: MIL-STD-498, MIL-STD-882E, and ISO / IEEE 12207

Today, worldwide militaries are slowly shifting to ARP4761/A and ARP4754A as summarized above and within prior dedicated chapters of this book. Most military aircraft however made prior use of non-ARP47XX standards including one or more of MIL-STD-498, MIL-STD-882E, and ISO / IEEE 12207. Frankly, it is this author's opinion that those other standards are good, and generally similar to the civil standards, particularly ARP4761 for safety. However, it is this author's experience that those other non-ARP47XX and non-DO-XXX standards are more prescriptive, and therefore more "subjective", therefore more difficult to apply and assess consistently. Below is an all-too-brief summary of the other primary standards which legacy military aircraft previously employed:

Mil STD 498 Summary.

The following briefly summarizes MILSTD 498:

- Military standard for weapons systems, including ground (unlike 178C's airborne only)
- 22 Data Item Descriptions which must be prepared for each project; these DID's prescribe document formats thus "how to"
- Software is composed of Units, groups of Units called CSC's, and groups of CSC's called Computer Software Configuration Item (CSCI) which is usually a single executable.
- While Waterfall isn't mandated, typical 498 processes look like a Waterfall
- Unlike 178C, 498 also prescribes management, risk assessment, and external ecosystem considerations devoid in 178C
- Like 178C, mandates standards for requirements, design, code, and tests (but 178C doesn't have testing standards).
- Unlike 178C, focuses upon hardware resources (memory, etc.) and higher priority software requirements
- Unlike 178C, gives details on incremental builds and real-world attainment of large complex programs including management and project risks.

- More black box testing; lacks 178C's dataflow, control flow, structural coverage, strict transition criteria, robustness testing, assessment, tool qualification, certification liaison

MIL-STD-882E Summary .

The following briefly summarizes MIL STD 882E:

- Strong on safety analysis (like ARP4761)
- Strong on risk analysis (like ARP4754A Safety Program Plan)
- Subjective risk assessment criteria
- Strong safety requirements, validation, verification, tracking
- Strong on CM
- Weak on software processes, design/code standards
- Strong on software and system testing

ISO / IEEE 12207 Summary .

The following briefly summarizes 12207:

- Heavily “borrowed” from DO-178C
- Like 178C, uses software “processes” not sequential “stages”
- Strong systems and software model – good for safety critical
- Copied from CMM then CMMI – strong on proofs and improvements
- Strong on software internals as well as black-box verification
- Strong on software design and code including timing and robustness;
- Weak on “objectivity”, - much more Subjective than 178C!!
- Strong on CM and evidence and V&V
- Weak on tool qual, DFCF, structural coverage to requirements, cert liaison, no HLR/LLR and less formal transitions.
- Much stronger than 178C on external processes such as management, metrics, business ...

GAP Analysis for Militaries converting to DO-XXX / ARP47XX.

Most military organizations and suppliers have established generally high-quality organizations and processes. When adopting DO-XXX and ARP47XX, they can reuse much of their existing processes, documentation, and artifacts. Often, they operate at a 60-70% DO-178 and 30-50% DO-254 adherence level without even considering DO-178/254. Therefore, when faced with the requirement to “comply with DO-XXX”, it is most cost-effective for them to do a Gap Analysis and simply address the gaps instead of starting over. This Gap Analysis assesses their current processes and determines the “gaps” compared to full DO-178 (or DO-254) adoption. A Gap Analysis typically takes 2-4 person weeks to perform by experienced DO-178/DO-254 experts, and can save years by maximizing re-use. And, unlike Military Standards which have strict requirements for document/artifact format/content, DO-178/DO-254 provides for greater latitude, hence the retention and reuse of existing items.

When this author performs a DO-178/DO-254 Gap Analysis for a military client, the following levels of “Gaps” are typically found within the audited organizations (where “0% Gap = 100% Compliance”):

- **SEI CMM Level 1 Organization: Gap = 70% – 90%**
- **SEI CMM Level 2 Organization: Gap = 50% – 75%**
- **SEI CMM Level 3 Organization: Gap = 35% – 60%**
- **SEI CMM Level 4 Organization: Gap = 25% – 40%**
- **SEI CMM Level 5 Organization: Gap = 20% – 35%**

There are two surprising facts from the above. First, even for CMMI Level 5 organizations, the gaps are still significant because CMMI does not include such 178C mainstays as two levels of software requirements, tool qualification, structural coverage (statement, DC, MCDC), extreme robustness testing, tool qualification, data/control flow and coupling analysis, etc. The second surprising fact from the above gaps is wide variation in CMMI Level 3 gaps; a gap of 60% for Level 3 only happens because the engineering more resembles Level 1-2 than Level 3.

On a DO-178/DO-254 individual activity basis, the particular gaps are typically as follows:

<i>CERTIFICATION ACTIVITY</i>	<i>% “GAP”</i>
<i>PSAC</i>	<i>80%</i>
<i>QA Plan</i>	<i>20 - 30%</i>
<i>CM Plan</i>	<i>10 - 20%</i>
<i>Development Plan</i>	<i>40 - 50%</i>
<i>Software Verification Plan</i>	<i>60 - 70%</i>
<i>Safety Assessment</i>	<i>80 - 90%</i>
<i>Requirements Definition</i>	<i>20 - 30%</i>
<i>Design</i>	<i>10 - 15%</i>
<i>Code</i>	<i>5 - 10%</i>
<i>Functional Test</i>	<i>5 - 10%</i>
<i>Structural Coverage Tests</i>	<i>90 - 100%</i>
<i>CM</i>	<i>10 - 30%</i>
<i>QA</i>	<i>50%</i>
<i>Tool Qualification</i>	<i>100%</i>
<i>Checklists</i>	<i>30 - 50%</i>
<i>Reviews</i>	<i>30 - 50%</i>
<i>Audits</i>	<i>30 - 50%</i>
<i>DER Liaison</i>	<i>100%</i>

Typical Gaps in New DO-178C and DO-254 DAL B Military Projects versus CMMI Level

3

DO-178/DO-254/ARP4754A Benefits on Military Projects.

DO-178 is not free, as cited above. However, DO-178 can be cost-effective, when understood and implemented properly, even on military projects. Why then are so many military organizations adopting DO-178/DO-254? Because there truly are actual benefits. The following describes the most commonly obtained benefits from DO-178/DO-254/ARP4754A for Military projects based upon the experience of this author’s success on over 150 aerospace projects:

- *Greater Supplier Visibility.* With DO-178 (and DO-254), the expanded artifact/review processes provide greater supplier visibility.
- *Greater upfront requirements clarity.* ARP4754A mandates safety and systems requirements covering all system-level functionality, safety, performance, derived, and interfaces with Hardware/Software. DO-178 mandates thorough and detailed software requirements, both high-level and low-level. Such detail, and the necessary discipline, force answers to be provided up-front instead of being deferred. Assumptions are drastically minimized. Consistency of requirements and their testability is assured. Iterations and rework due to faulty and missing requirements are greatly reduced.
- *Fewer implementation iterations.* Implementation and code iterations, or churn, are the bane of software/hardware engineering. In many cases, 10, 20, and even 30 versions of evolving code files exist on new products. Nonsense. Code and logic should be largely correct the first time it is written and should not require dozens of updates to get it right. Code should be reviewed by analyzing implementation versus documented requirements.
- *Decreased single-point project failures.* Software is an art; artists resist documenting their work and subjecting it to common development standards and peer reviews. Without standards, discipline, and modern software engineering principals, software teams devolve into a group of loosely structured rogue artists; these artists are highly valuable, creative, and talented persons. But, the loss, or deficiency, of any such artist for any reason is catastrophic to the team. Unless their work is documented, understood, and consistently applied as for the other artists. DO-178C greatly reduces the possibility of such single-point project failures.
- *Improved management awareness of true schedule status.* How many software projects report a “99% Complete” status week after week? How is software progress measured? How can management truly ascertain completion status of software? The answer to all these questions is via modern and accurately detailed management techniques built around DO-178. The provision for insight, traceability, and

accurate status on design, development, testing, integration, and reviews is found in DO-178.

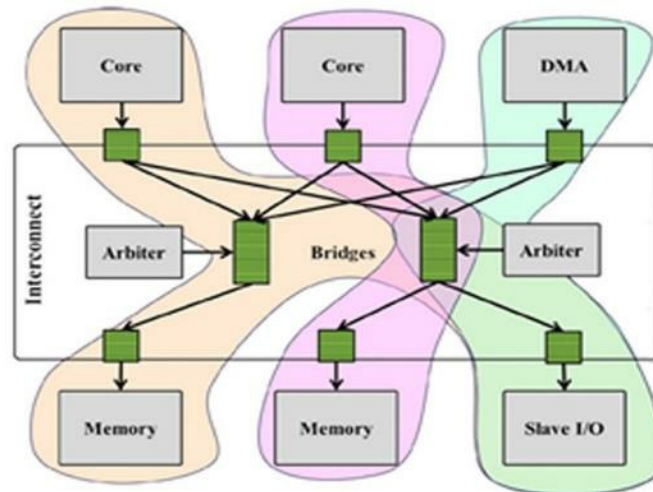
- *Greater consistency within software.* Software is like a chain: only as strong as its weakest link. Software that is 99% correct is 1% incorrect, which means it is unsafe. The weakest software module, or software engineer, is on the critical path of software safety. All software must be consistent per its level of criticality and DO-178 enforces such.
- *Fewer defects found during integration.* Integration can be a lengthy iterative process where major defects requiring design changes are revealed and fixed. Not with DO-178, where integration is typically 50-75% faster than non-DO-178 environments.
- *Improved reusability.* Via thorough and consistent documentation required by DO-178C, modularization, enforcement of documented modern engineering principles, and reviews to ensure all the above was achieved, re-usability is greatly improved. In software, re-usability is the holy-grail. But the reality is that unless a software component is at least 80% re-usable (e.g. unchanged), then it is quicker and less risky to simply start from scratch. And most software is less than 50% “reusable”. With DO-178C, and enforcement of design/coding standards, coupled with independent reviews and traceability, most modules should be at least 90% reusable.
- *Easier regression testing.* You build your product on the assumption that it will be successful, and therefore have a long life. And you know your software will evolve through new applications, installations, and versions. Regression testing can be expensive if extensive or manual; DO-178 provides thorough traceability to determine which modules need changes or analysis, and corresponding retesting.
- *Improved hardware integration.* Integrating software onto its target hardware system is typically challenging for embedded systems: the development environment is quite removed from the target environment and implemented via different engineering teams. DO-178 mandates that software components which could be affected by hardware be tested on the hardware, e.g. hardware/software interfaces, interrupts, timing,

board-level components, BSP/RTOS, etc. And the improved determinism and quality of all these components via DO-178 belies improved hardware integration.

- *Improved System/Safety Focus.* Military projects increasingly follow civil aviation's mandate to consider ARP-4754A for System considerations and ARP-4761/A for Safety. Where previously militaries often followed MIL-STD-882E (2012), DEPARTMENT OF DEFENSE STANDARD PRACTICE SYSTEM SAFETY, today they are increasingly harmonizing with the civilian system/safety counterparts ARP4754A and ARP4761/A just as they did for software and hardware, DO-178C and DO-254, respectively. By applying the Systems/Safety requirements up front at the Aircraft then the Systems level (and then iteratively and concurrently) continuously throughout the project, the overall aviation development and safety adherence are improved.

Multi-Core Processors for Avionics Via CAST-32A (With IMA: DO-297)

By Mark Brown, Senior Architect, Lynx Software & Vance Hilderman, CTO AFuzion Inc.



Multi-Core Processing (MCP) and CAST-32A

Newcomers to aviation development and certification often have a pre-existing opinion that the field is both static and staid. And frankly, simply looking at the 10-20 year update frequency of individual DO-XXX and ARP47XX guidelines, that opinion would seem to be correct. And Covid-19 is simply the flu ... hardly!

Computing technology, with the possible exception of recent Biotech advances, is perhaps the most rapidly advancing field in the history of our planet Earth. It should be no surprise that aviation's computing technology is likewise rapidly advancing. Yes, civil aviation computing will always lag behind both the bleeding, and even leading, edge. Public safety mandates such. Prior chapters in this book dealt with topics such as Model-Based Development, Object Oriented Technology, and advanced software tools; none of these existed four decades ago but are commonplace today.

Humans rarely have unanimous consensus on any topic with the possible exception of computing horsepower: we need more of and more it,

continually. In the past, semiconductor manufacturers simply engaged in continuous electrical signal distance shrinkage and optimized single processors to achieve greater processing speeds. But like a finite water supply, eventually the ability to add greater throughput evaporates. Such is the case with Yesterday's single core microprocessors: the best way to achieve greater processing power was to combine multiple coordinated processors within a single processing unit, known as a Multi-Core Processor (MCP).

CAST-32A presents the coordinated position of avionics certification authorities regarding MCPs. While today's aerospace ecosystem could benefit from the use of MCPs, when CAST-32A was published FAA/EASA had not yet devised a means to obtain certification credit for safety-critical software deployed to an MCP. Toward that end, the CAST-32A position paper identifies topics of concern that could impact the safety, performance, and integrity of DO-178C aviation software deployed to MCP(s).

For each topic, this paper provides a rationale that explains why these topics are of concern and proposes objectives to address the concern. (CAST-32A, "Purpose", p. 3)

Since relevant avionics software certification documents (DO-178B/C and ED-12B/C) were written before MCPs were used in civil aircraft, those certification guidelines can only address software executing on single-core hardware. The Certification Authorities Software Team (CAST) is an international team of aviation experts who clarify and harmonize the aviation development ecosystem. Their CAST-32A position is that MCPs could credibly deliver size, weight, power, and cost (SWaP-C) advantages and that today's aerospace equipment suppliers are interested in using MCPs in their systems.

The consumer device world has fully embraced MCPs and many of the devices used daily by the readers of this paper contain MCPs. Some in fact predict the obsolescence of single-core processors (SCPs) altogether.

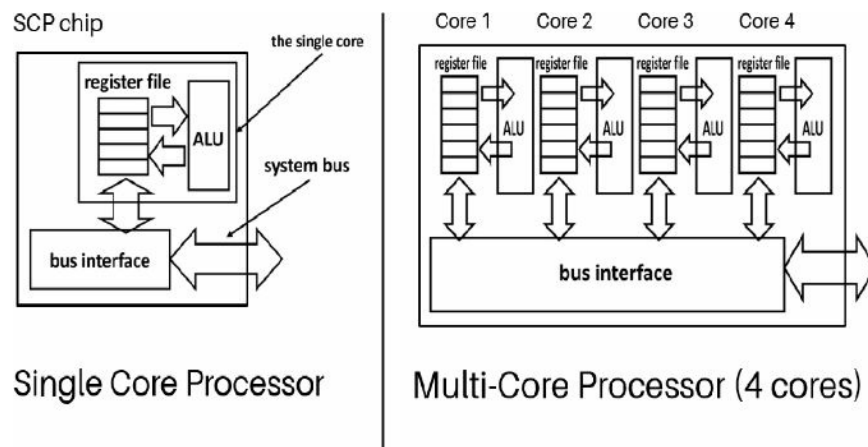
Tomorrow's avionics will most certainly contain more sophisticated avionics, meaning greatly expanded processing power. MCPs are a major solution to this rapidly expanding need for enhanced computing architectures and

processing power; therefore, the aerospace industry in general must consider how best to utilize MCPs in future designs. But how can MCP challenges be overcome?

Before describing MCP (and CAST-32A's) topics of concern, which emphasize partitioning and its degradation by interference, first consider the background influencing the use of MCPs. Engineers and managers who already anticipate using MCPs as hardware targets for their next generation of software will benefit from considering these influences for future designs.

Single Core Vs. Multi-Core.

The following simplified diagram depicts essential differences between single-core and multi-core:



Single Core vs. Multi-core Diagram (Simplified)

As shown above, each single core (left hand side) is comprised of a set of registers and an associated Arithmetic / Logic processor Unit (ALU). Multithreaded cores (not shown) may be designed after duplicating the most highly used parts of a single core, e.g., by adding a second register file. In true MCPs (shown above, on the right hand side), cores are tiled to create a larger set of cores. Typically, both SCPs and MCPs use a bus interface to access an increasingly rich and complex network of “uncore” semiconductor and/or electronic peripheral resources.

Different developers assign varying reasons for utilizing MCP, but common advantages cited are:

- Higher performance

- Simultaneous execution
- Less power and heat per individual instruction executed
- Use of multithreaded applications
- Supports Moore's Law (see details herein) • Access to "latest" and often best-in-class commercial technology

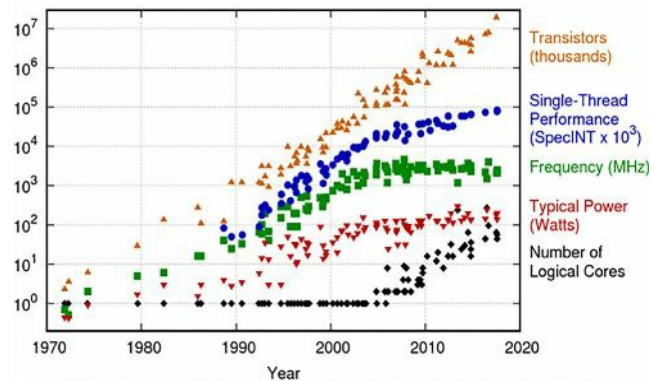
House, Apartment, or Hotel?

PC, server and mobile markets using symmetric multi-processor (SMP) operating systems have greedily gobbled up commodity MCP parts available since 2005. Enterprise server rooms simply scaled up, completing heavier workloads faster. In contrast to these markets, real-time deadlines and safety concerns characteristic of DO-178C-certified software have delayed the adoption of MCP for avionics software. The topics of concern presented in CAST-32A get at the nub of the difference between SMP operating systems, which have readily scaled up to utilize MCPs, and real-time operating systems which have not.

These concerns are emphasized in many other FAA and RTCA publications; for example, a reference model for real-time processing is given in FAA AR-05/27. In that model, real-time avionics systems are seen as computing a "control law" which takes inputs from sensors and gives outputs to actuators on a fixed periodic basis. When the actuator controls a flight surface or some other safety-critical aspect of the aircraft, missing the deadline can have catastrophic effects. In these contexts, the most challenging problems facing MCP deployments involve protecting, or partitioning, the software responsible to meet these safety-critical deadlines and ensuring the determinism of unchangeable manufacturer logic designed into the MCP.

In an enterprise context, tasks, processes and threads are encouraged to move out from their home processor more frequently – at a rate which may be compared to Moore's Law. In contrast, re-hosting real-time embedded systems raises safety concerns and certification costs which must be considered afresh whenever avionics software is deployed to a different model of processor core. As a rule, in avionics (and safety-critical) contexts, software requiring deterministic real-time deadlines is more tightly bound to a specific processor core and target board. Because of this, Moore's Law both

helps and hurts real-time systems. A plot of Moore's Law (below) helps to frame these concerns in terms of exponential growth over decades:



42 Years of Microprocessor Trend Data

A useful metaphor to frame this discussion may be found by considering how to re-host a real-time, safety-critical task from its own dedicated SCP to a “new home.” Is the new home a (1) permanent dedicated-to-use structure, like a house, (2) a shared-use structure, like an apartment, or (3) a time-shared room, like a hotel?

Is CAST-32A Significant Today?

In 2014, the FAA released CAST-32, which is obsoleted by CAST-32A. At the time, independent industry experts, such as David Arterburn, Director of Rotorcraft Systems Engineering and Simulation Center at the University of Alabama at Huntsville, downplayed the short-term prospects of multi-core processing, pointing to the difficulty of understanding and predicting complex interactions within quad-core chips. Arterburn was at the time compiling 36 studies of various Army Aviation PEO-funded working groups. V. H. Dova reports in his evaluation of the technical readiness levels of future Integrated Modular Avionics (IMA2G or 2nd Generation IMA):

‘Arterburn downplayed the significance of the CAST-32 paper, asserting that no multi-core chip installation has actually achieved official airworthiness. He was quick to point out however that the issue will soon come to a head. “Within five years” he predicted, “you won’t be able to buy a commodity single-core processor” (D. R. Arterburn, personal communication, January 22, 2015).’ (as reported in Dova, 2015)

The graph of Moore's Law (previous page) supports Arterburn's prediction. Not only are commodity MCPs on the rise as SCP production declines, but the red triangle plot of Transistors will inevitably hit a wall stemming from a limit of physics (the minimum size of a semi-conductive crystal) and a cost-benefit limit (the commodity price per transistor). Leading voices in this prediction include Bob Colwell, formerly Intel's IA-32 chief architect and Intel Fellow (as reported in MIT Technology Review). Gordon Moore concurs (as reported by Rachel Courtland for the IEEE).

While exponential growth in the number of transistors will cease, the number of logical cores per microprocessor package (or "chip") has grown exponentially since before 2010. It is this new MCP trend, shown in the graph above as the plot of black diamonds, which CAST-32A addresses.

While CAST-32A "proposes objectives to address the [MCP] concern," the bad news is that satisfying these objectives remains as an obligation for those who choose to use MCP chips. The good news in CAST-32A is that satisfying these objectives will enable the future use of MCPs in real-time and safety-critical aeronautics contexts.

The significance of CAST-32A, therefore, will be seen if it enables the use of MCPs in safety-critical contexts. Given our metaphor, the question becomes how to move hard real-time tasks out from their permanent, dedicated-to-use SCP homes and into new MCP homes. At the same time, CAST-32A may be seen as enabling the consolidation of soft real-time and non-essential processes to share the same MCP as apartment or hotel dwellers. More than ever before, it is likely that a partitioning architecture will be needed to ensure that future consolidations of airborne software tasks continue to reliably meet their real-time deadlines on MCPs. When MCPs are used to consolidate software which had previously been deployed to separate hardware processors - i.e., to multiple chips - physical separation which had been previously assumed (and more readily proven) now needs to be assured.

Moving Software To a New Home.

Engineers within the broader field of embedded systems can affirm yet another crucial market force affecting future chips. The decision between marketplace-dominant instruction architectures, e.g., PowerPC, Intel, or ARM, itself opens

Pandora's Box. Since about the time of the release of the Apple iPhone® in 2007, worldwide shipments of smartphones now far outstrip the PC and server computers to which Moore's Law had been applied for decades. Viewed as a broad influence on the semiconductor history, iPhone® and the ensuing popularity of smartphones in the past decade has catapulted these embedded systems to become the world's most widely deployed type of computer system. Deciding between commodity PowerPC, Intel or ARM chips tends to bleed over into peripheral concerns associated with input-output and co-processor connectivity conventions associated with supercomputer, PC/server, and mobile deployments.

So CAST-32A must also be considered in context of the market influence of the SOC. Popular PowerPC, Intel, and ARM chips use distinctive system-on-a-chip (SOC) designs and an increasingly large assortment of system-level resources. That is, in contrast with an array of identical processor cores (MCP), most commodity chips today also provide a gamut of heterogeneous communications resources ranging from caches and on-die DRAM to input-output devices and high-speed interconnects (which can connect either device farms or additional processing cores). Today it is difficult to find a commodity chip which does not include uncore system resources, which may include power management, memory management, debug and clock management, interrupt controllers, on-chip input-output devices, etc.

Issues which must be addressed from a safety perspective include both processor-core MCP as well as the uncore system resources. Granted the broader safety concerns presented by the concept of interference, CAST-32A addresses both multiple processor cores and additional system-level hardware blocks (often referred to as Intellectual Property [IP] cores) present on the chip which could give rise to interference. In this way, the summary-level concern raised in CAST-32A is the threat of interference to some hard-real-time software component. As the complement to its concern about interference, CAST-32A also defines "Robust Partitioning," which provides measurements to demonstrate that a partitioning imposed upon the MCP deployment is robust against interference threats with respect to both time and space.

Whether or not today's engineer chooses to move a hard real-time software component to a new processor home in a commodity MCP chip, the threat of

deadline interference to real-time software is the primary safety concern.

Whereas pre-2005 commodity chips could be expected to provide a physically isolated single processor core, titanic forces commonly associated with Moore’s Law have increased both the number of IP cores and the significance of interference threats to real-time deadlines. Today, moving a hard real-time process out from its legacy home on a single-core processor and into a new home on a multi-core processor SOC requires more attention to interference concerns than ever before.

MCP Interference Concerns.

CAST-32A organizes MCP concerns into a set of questions. These questions are answerable, in view of the 2017 publication of FAA TC-16/51. Measurable answers to the CAST-32A concerns will define a partitioning architecture which is sufficiently “robust” against all identified interference threats. The CAST-32A “topics to address,” or questions, are summarized in the following table:

CAST-32A Question / Topic To Address	Document Contents (Summary)
MCP_Planning_1	Planned MCP usage
MCP_Resource_Usage_1	MCP Critical Configuration Values
MCP_Resource_Usage_2	Critical Configuration error detection and correction (EDAC)
MCP_Planning_2	Shared Resource List
MCP_Resource_Usage_3	Interference Channels List
MCP_Resource_Usage_4	Worst Case Resource Usage
MCP_Software_1	Worst Case Execution Time, per software component instance
MCP_Software_2	On-chip data and control communications resources
MCP_Error_Handling_1	Within-MCP Failures and Effects on Safety Objectives
MCP_Acc_Summary_1	Augmented SAS

Table 1: Summary of CAST-32A “Specific MCP Topics to Address”

This set of questions relies upon a prior definition of “resource,” which in CAST-32A’s usage depends on “Robust Resource Partitioning.” It also depends on “Robust Time Partitioning” for Worst Case Execution Time and gives several other definitions that will frame valid answers to the CAST-32A questions. These are discussed below. While CAST-32A does not constitute official policy or guidance from any certification authority, its definitions for “robust” resource- and time-partitioning “should be discussed with the appropriate certification authority” during certification projects.

(This “NOTE” is repeated as a footer for every page in CAST-32A.) However, what is also clear is that no other guidance on how to develop system software for multi-core exists, at least not defined and expressed by the certifying bodies. Hence CAST32A is the only viable way forward.

Robust Resource Partitioning.

What is a resource? By analyzing Table 1 (above), we find that half of the rows depend on the definition of “resource” as used in CAST-32A (i.e., “MCP_Resource_Usage” questions RU1, RU2, RU3, RU4, the “Shared Resource List” of P2) and that all of the rows have at least an indirect dependency on the definition of “resource” (e.g., the on-chip data and control communications of S2 can be seen as resources).

First, note that a resource is a partitioned resource. From the perspective of some hard real-time software component running within an MCP, how does CAST-32A define “Robust Resource Partitioning”? As follows:

Robust resource partitioning is achieved when:

- Software partitions cannot contaminate the storage areas for the code, I/O or data of other partitions.
- Software partitions cannot consume more than their allocations of shared resources.
- Failures of hardware unique to a software partition cannot cause adverse effects on other software partitions.

NOTE: Software that provides partitioning should have at least the same DAL as the highest DAL of the software that it partitions.

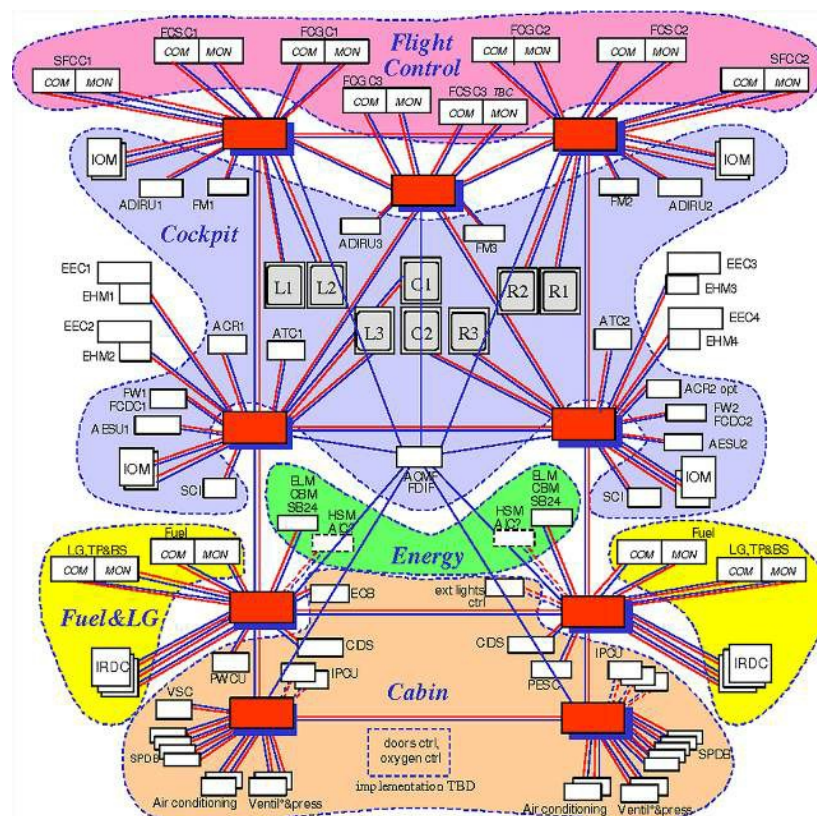
The definition was “adapted from DO-248C / ED-94C and DO-297 / ED-124.” In other words, CAST-32A was able to reconcile its MCP certification guidance with already-published certification standards which address partitioning architecture. Granted that partitioning is in place for a resource, DO-297 further defines resources as described in the following section.

Resource Partitioning: for an IMA or for an MCP?

DO-297 defines Integrated Modular Avionics (IMA) abstractly as a “shared set of flexible, reusable, and interoperable hardware and software resources that, when integrated, form a platform that provides services” (Appendix E,

Glossary), or more briefly, a set of resources. Specifically, a resource is defined as “Any processor, memory, software, data, or object or component used by a processor, IMA platform, core software, or application. A resource may be shared by multiple applications or dedicated to a specific application. A resource may be physical (a hardware device) or logical (a piece of information).” CAST-32A seizes upon this abstract definition and makes it fit in the context of MCPs. Taken together, “robust partitioning” from DO-248C and “resource” from DO-297 comprise the Robust Resource Partitioning concept within CAST-32A.

By drawing its definition of resource from DO-297, the CAST-32A position anticipates certifying an IMA-on MCP architecture. Considered abstractly, CAST-32A takes a conservative step, but with respect to modularity, network and replacement the effects are large. For examples of IMA-on-SCP resources, consider the following diagram from “The Airbus Approach to Open Integrated Modular Avionics (IMA)” (H. Butz, 2007). Resources are accessible to software executing within computers (shown as white rectangles):



The diagram above portrays the “Open IMA” used by the Airbus A380, and includes an AFDX Ethernet network connecting many SCP, avionics standard (ARINC 600), core processing and IO modules. Shown in the diagram are colorful shapes which present the A380’s logical IMA Usage Domains (UDs). UD’s include:

- Flight Control
- Cockpit
- Energy
- Fuel and Landing Gear (LG)
- Cabin

Within the UD shapes, white rectangles show SCP compute modules, and lines indicate connectivity to the AFDX network (nine redundant Ethernet switches are shown). As partitioned, IMA resources include AFDX Ethernet time slices and subsystem-specific sensors and actuators, as well as general compute resources such as memory and processor time slices. Partitioning of resources, including memory and processor time slices, leverages an RTOS to provide time and device partitioning.

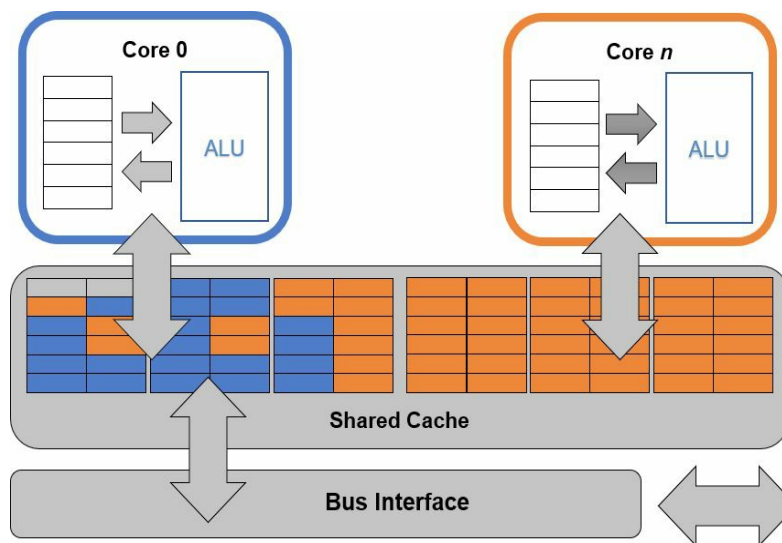
Integrated Modular Avionics (IMA) on a Chip.

While this new IMA network must provide inter-partition communications within the MCP, it will not be required that any partition leverage AFDX Ethernet switches, gold-plated ARINC 600 connectors or even Ethernet MAC or PHY hardware blocks. Instead, simply sharing pages of DRAM between partitions could satisfy CAST32A’s Shared Resource Capacity verification activity (MCP_Software_2).

Granted the relative throughput of switched Ethernet communications compared to on-chip memory accesses, a shared-memory interface could improve throughput by several orders of magnitude. Yet at the same time, shared memory interfaces may introduce significant nondeterminism when cache memories are shared.

A processor cache provides a mechanism to significantly increase CPU performance by temporarily holding copies of content in the main memory

which are predicted to be accessed in the near-term. This enables the CPU to operate on data that is in use frequently without having to perform a slower memory access via the much slower memory bus. Most CPUs have a hierarchical composition of caches with different levels, for example Level 1 to Level 4 cache (L1 – L4). The low-level caches typically are dedicated to a specific core, but the higher-level caches are shared between the processor cores to enable multiple access and even higher throughput. This means the CPU has to manage processor/cache synchronization issues when a shared cache is used. Obviously this is a significant resource interference issue: the state of a shared cache could be affected by an erroneous or non-synchronized usage thus negatively impacting the performance of a safety critical partition. As an example, any large transfer through a shared memory interface affected by a cache, specifically, one that is shared with an application or core not participating in the transfer, could experience significant cache-thrash while competing with the shared memory transfer. The diagram below illustrates one such scenario affecting MCPs:



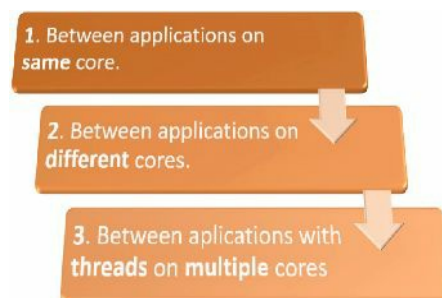
Interference arises when unrelated applications compete to pin their memory contents into that same shared cache. As shown above, a Shared Cache has been added to an MCP. Core 0 is losing its competition with Core n to populate this Shared Cache, and as a result, Core 0 will experience less determinism and greater latency when accessing memory locations due to cache misses. Thus MCP_Software_2 may represent both the “killer app” and the Achilles Heel associated with the MCP questions raised by CAST-32A.

Robust Partitioning Vs. MCP Interference.

CAST-32A compliance requires answering the following question: Given an IMA context, can an MCP be used to safely consolidate partitioned resources from multiple SCP compute modules? Relying on a bevy of prior definitions, CAST-32A formulates the following definition for its intended Robust Partitioning:

“MCP Platform With Robust Partitioning: an MCP platform that complies with the objectives of this document and provides Robust Resource and Time Partitioning as defined in this document, not only between software applications hosted on the same core, but also between applications hosted on different cores of an MCP or between applications that have threads hosted on several cores.

Granted this definition of Robust Partitioning for an MCP platform CAST-32A will require consideration of any hardware-supported application parallelization. Specifically, designers must be explicit about which applications are allocated to which hardware cores or threads, if applicable. Addressing these concerns will require software designers to consider the following interference scenarios, presented in ascending order of difficulty:



As implied by this hierarchy (above), efforts to obtain robust partitioning can be simplified by minimizing or eliminating interfaces and communications between applications. However, a concern arises whenever an application (or a time slice allocation to an application) migrates from one hardware core (or thread) to another. Since SMP operating systems treat all cores (and all time slices of cores, with respect to scheduling) as symmetrically interchangeable, a byproduct of this SMP convention is that knowledge about an application’s allocation to one core or another is hidden from the software developer by the OS. In the same token, an SMP OS is generally free to migrate an application

to a new core after preemption.

While an avionics software designer may want to minimize cases where an application's threads could be allocated among multiple cores (see level 3, above) an SMP OS may not provide the best tools to force asymmetric distinctions onto assumed-symmetric cores. Taking this concern to an extreme, some avionics designs go so far as to disable all cores except one on an MCP. Doing so ensures application-to-core binding, while still realizing many benefits of an OS. But the main benefit of MCP-based designs is to harness the greater computational abilities of simultaneous processing and increased computing power, which is lost by disabling all except one core. Therefore, more satisfactory MCP solutions will include both assured application-to-core allocations and the assurance of Robust Partitioning as defined in CAST-32A.

By tracing several definitions back to DO-297 (2005), CAST-32A sheds the burden of defining from scratch a partitioning architecture, its partitioned resources, and the configuration, integration and other goals of an IMA. Instead, CAST-32A advances its questions specifically about MCP platforms, granted that "robust partitioning" is already defined.

Life-cycle Phase	CAST-32A Topic To Address	Proposed Interpretation
Plan	MCP_Planning_1	Resources List: Identify the MCP chip, the number of active cores, their clock frequencies, their caches, and any dynamic features of/affecting the processing cores. List the applications, and the components comprising the partitioning architecture including any RTOS, ARINC653 and/or hypervisor components used.
Plan	MCP_Planning_2	Shared Resources List: Document a plan which includes: - A list of shared resources, which includes plans for how to verify their use, how to avoid / mitigate contention, and how to prevent their exhaustion due to demands placed on them by planned software or hardware components. - A list of hardware dynamic features that will be active, and a plan for how they will be used.
Plan	MCP_Resource_Usage_1	Partitioning Configuration: Document the configuration settings data which control the resource and time partitioning Identified in (P1) and (P2)
Plan	MCP_Resource_Usage_3	Interference Analysis: Identify Interference channels that could permit Interference to affect the software applications hosted on the MCP cores. - Include consideration of shared memory, cache, interconnect or IO hardware resources. - Describe the verification approach planned for any mitigations chosen by the applicant. - Omit interference channels which cannot affect software applications, granted the intended final configuration for the system. - Include any interference channels discovered at any time during the project. - Omit this objective and its analysis in certain IDAL C cases.
Build / reuse	MCP_Resource_Usage_2	Configuration EDAC: Develop, document and verify means to protect the Critical Configuration Settings data for the MCP, including FDAC.
Build / reuse	MCP_Error_Handling_1	Failure Analysis and Mitigation: Identify, plan, design, implement and verify means to detect and handle failures which may occur within the MCP. Contain the effect of any failures within the equipment in which the MCP is installed (may use a "safety net" within this equipment).
Verify	MCP_Resource_Usage_4	Resource Capacity: Verify that the demands upon the resources of the MCP and on interconnects do not exceed capacity, given the (P1) partitioning, the intended final configuration settings, and the identification of Worst Case scenarios.
Verify	MCP_Software_1	Application WCET: Verify that all application functions correctly and have sufficient time to complete their execution, given the intended final configuration including its time partitioning.
Verify	MCP_Software_2	Shared Resource Capacity: Verify correct functioning for each application, given all configured data and control couplings between itself and all connected software components executing on the same or different cores of the MCP.
Report	MCP_Acc_Summary_1	Accomplishment Summary: Append the answers to these questions into the DO-178C Software Accomplishment Summary (SAS).

Acknowledging how CAST-32A borrows resource-partitioning concepts from an IMA context allows CAST-32A to be more readily interpreted. A realistic query would consider how a new MCP chip could consolidate a plurality of prior-generation-IMA SCP compute modules (e.g. ARINC 600 pluggable) to further reduce SWaP-C. Given this context, one could interpret the CAST-32A questions as follows (see diagram on following page).

In such an interpretation, CAST-32A “topics to address” are allocated into system development lifecycle phases based on analysis of CAST-32A descriptions of the effort required by the task. The intent is not to alter the task as stated in CAST-32A, but rather, to help evaluate when in the system development lifecycle these tasks must be performed. Additionally, the proposed interpretation suggests a goal, i.e., what an engineer and manager must target in order to complete the task. Based on these interpretations, most CAST-32A tasks must be inserted as new planning-phase activities, and also as post-coverage verification activities.

Challenges from CAST-32A Interference Analysis.

CAST-32A can be interpreted as organized by lifecycle phase, calling out new planning, implementation, and integration-verification tasks (see Table 2 above), thereby mimicking DO-178C. Engineers and Managers who intend to certify one or more applications under CAST-32A for MCP must allocate work and staff to address these tasks, where the primary effort may be expected to occur after coverage-based verification has been completed, and performance-based testing can begin. (Remember, “coverage” means first “requirements coverage” and then for DAL C and above added “structural coverage”.) FAA-sponsored research in TC-16-51, which was used to improve CAST-32 to become CAST-32A and now serves as an informative supplement to CAST-32A, identifies the effort this way:

“An interference analysis is, at first, a matter of performance assessment. Assuming that, ideally, the processor has no failure modes triggered by interference channels, some interference channels will have a bounded and acceptable impact on software execution time. ...” (TC-16-51, p.11)

While CAST-32A nominally anticipates project efforts to complete an interference analysis, FAA TC-16-51 presents a much more detailed report

on what must be analyzed to ensure that these anticipated efforts will be successful. Whereas traditionally top-down analyses of hazards and failure modes have been required to assure safety in the aeronautics industry, TC-16-51 advises that after a top-down analysis has assigned IMA applications to partitions called Usage Domains (UDs) as in the Airbus Open IMA (see above), a bottom-up interference analysis must also be performed. Specifically, TC-16-51 asserts:

The application of the top-down approach is a necessary step to set bounds on the investigation of sources for non-determinism [interference] as part of the bottom-up approach (discussed in section 6.1.2). While [the top-down approach] facilitates industrial and certification processes ... this approach is not sufficient. ...The top-down approach provides the selection criteria for the MCP [chip] and for selecting the IP to be activated or configured within the MCP [solution]. This configuration defines the UD of the MCP in which determinism is [to be] guaranteed. (TC-16-51, p. 29, emphasis added).

Results from TC-16-51's proposed new bottom-up approach may be summarized as answers to CAST-32A's questions, or then condensed into the engineer's answers to MCP_Software_2. Analyzing the interpretation of CAST32A in Table 2, the authors suggest that the total set of questions raised by CAST-32A will produce a measurement framework useful for observing relevant interference channels. Measured interference results, first for MCP_Software_1's WCET in view of dedicated resources and then for MCP_Software_2's non-interference in view of both dedicated and shared resources, will roll up into a final score for bottom-up-measured interference on the MCP.

Describing the bottom-up measurement techniques and analyses for various hardware resources presented in TC16-51 and its references is beyond the scope of this whitepaper. Rather, at the CAST-32A level of discussion, it is noted that the bottom-up measurement of hardware-based interference channels may be classified as follows, and then summed up for resource in each UD partition:

Acceptable	• Bounded: The interference channel has a bounded impact, so that an interference penalty can be assessed. • Known-Valid: The measured interference penalty copes with the equipment's functional domain requirements.
Unacceptable	• Bounded: The interference channel has a bounded impact. • Known-Invalid: The interference penalty that was assessed is too high. • Mitigation required: Changes must be made to meet performance requirements.
Unbounded	• Unbounded: No interference penalty could be assessed on this interference channel. • Unknown-Invalid: Its impact on software execution time is not properly known. • Mitigation required: Changes must be made to assure performance requirements.
Faulty	• Unbounded-failed: The interference channel has triggered a failure mode within the processor or MCP. • Known-Invalid: Further investigation in collaboration with the manufacturer is required and formalized through a safety analysis. • Mitigation requested: Correction by the manufacturer is requested.

Interference Analysis "Tags" Applied to Observed Hardware Channels

How difficult might it be to complete this interference analysis, for all interference channels, for all resources in an IMA UD? A formula (TC-16-51, p. 56) and a table demonstrating exponential growth in interference paths for examples shows that completion of this task is unreachable except for very simple architectures (TC-16-51, p.57). TC-16-51 recommends what may be interpreted as all "possible simplification in the strategies to guarantee determinism at the MCP level" (p. 30). Without such strategies, completion may be difficult or impossible. "A major challenge of conducting interference analysis is to reach a compromise (in time and cost) of the test campaigns and the need to have a trustworthy coverage of interference situations" (TC-16-51, p. 11). In brief, selecting a very conservative partitioning design may be the most successful approach for certifying an MCP system under CAST-32A as currently informed by TC-16-51.

Benefits from Using IMA.

Engineers and Managers who intend to certify one or more applications under CAST-32A for MCP must grapple with effects on the development lifecycle similar to other IMA-based projects. Essentially, there is an upfront price to

pay for downstream benefit, e.g. “no free lunch”. However, while the subsequent lunch may not be free, it should be greatly discounted: benefits of improved IMA integration, re-use, and quality should outweigh the added upfront costs. One way to think about the benefits to using IMA is to consider what can be certified under FAA AC 20-170. Consider the chapter titles to demonstrate what can be certified and approved under FAA rules for IMA, in contrast to system-only certifications:

- Chapter 2 Acceptance and Incremental Acceptance
- Chapter 3 Obtaining an IMA Component Acceptance Letter• Chapter 4 Reuse of IMA Components
- Chapter 5 Configuration Management on an IMA System
- Chapter 6 IMA Recovery Features

The benefits of IMA may be seen by considering this subset of the AC-20-170 table of contents. These benefits include reduced certification efforts due to incremental acceptance, component acceptance, and reuse of accepted components. These cost-benefits can accrue over years, over multiple subsystems, components, suppliers and/or aircraft systems.

Considerations for Using IMA on MCP.

Because definitions for IMA modules, components, and applications all include both hardware and software, once they are certified, they can enjoy the benefits of modular hardware-based replacement. This compares favorably to FAA certification and approval practices for purely software-based approaches to modularity. A practical implication of using a hardware platform for IMA modules is that any specifically instantiated IMA system’s modularity resolves to its selected standard for plug-ability. For example, in the Airbus Open IMA, its standard connector is a gold-plated ARINC 600 plug. Taking these practical effects into account, DO-297 and AC 20-170 IMA certification and acceptance rules could be interpreted as emphasizing hardware standards for connectivity, e.g., AFDX Ethernet networking or ARINC 600 plugs. However both AFDX (ARINC 664) and MILSTD-1553 (ARINC 429) have been used for IMA architectures, and a NASA comparison of communications architectures for aerospace modular avionics

systems surveys a variety of communications protocols which may be used for IMA communications systems (NASA/ TM 2006-214431).

Future MCP-based generations of IMA architecture should take note of these practical considerations. IMA connectivity standards must change when IMA applications are re-hosted on MCP because the Robust Partitions in CAST-32A are no longer line-replaceable networked compute modules, but are rather their virtual machine equivalents, e.g., only one core or time slice of a core within an MCP, and its partitioned resources. How does one time slice of a core communicate with another time slice on the same MCP? Or perhaps more importantly, how can the beneficial plug-ability of IMA components be retained when ARINC 600 physical plugs are subsumed within an MCP chip? While implementation-specific standards may become obsolete when IMA components are re-hosted to MCPs, the need for standards-based communication and plug-ability (supporting line replacement) will remain fully intact.

CAST-32A Impact .

For engineers and managers, CAST-32A and its supporting TC-16-51 open the door to future multi-core and many-core chips now appearing on the horizon of predictions from Moore's Law. It is expected that traditional avionics applications will "move out" to find a new home in MCPs.

For the broader avionics industry, the CAST-32A position may harness raw compute power to do useful work. Successes in the commercialization of COTS chips pressurize the need for MCP avionics, as anticipated in CAST-32A. Yet due to the challenging questions posed in CAST-32A and the difficulty of avoiding risks from MCP sources of non-determinism, as seen in TC-16-51, the current FAA consensus appears to create a metaphorical steam engine such that Robust Partitioning of Integrated Modular Avionics is the piston.

The questions, "What are architectural strategies?"

(section 2.4) and "What is a partitioning architecture?" (section 2.4.1) already present in DO-178C may now be seen as converging in CAST-32A, including its reliance on IMA. In view of historical IMA successes, both in military and civilian aircraft, and seen also in NASA space systems, CAST-

32A's reuse of DO-297 IMA definitions supporting Robust Partitioning seems warranted. If the future avionics and aeronautics industry converges on MCP IMA platforms, CAST-32A will be seen as the tipping point.

Aviation Development/ Certification Costs vs. Benefits

By Vance Hilderman

Reviewed by:

- Mr. Carmelo Tommasi

Introduction.

“DO-178, 254, 278 are the worst standards in the world ...except for all the others.” (Vance Hilderman, 2004)

The above rewrite of Winston Churchill’s famous quote about Democracy has some truths: the DO-XXX guidelines are the benefactor, or bane, of aviation projects the world over. Many users complained of added costs associated with DO-178 since its creation forty years ago, and later versions and sibling guidelines did nothing to reduce costs as they all added even more rigor. Truly, DO-XXX and ARP47XX are never cheap, certainly not on the first project. And in clear cases outlined herein, DO-XXX and ARP47XX will increase costs by a minimum of 20-40% beyond what great engineering companies would spend otherwise, on the first attempt. But are DO-XXX and ARP47XX really “too” expensive? Don’t they actually reduce costs longer term for companies who were developing smart? Do they reduce long-term costs at the expense of increased initial development cost? Will they improve safety and reliability and, if so, to what degree? Exactly what benefits are received from complying with DO-XXX and ARP47XX? Can companies reduce certification costs without impacting quality? These important questions are addressed and answered below.

DO-XXX and ARP47XX have increasingly evolved into the de-facto standards for virtually all forms of civil aviation except for experimental aircraft. In the past decade, DO-178B and now DO-178C/DO-254 are mandatory for most Military avionics. The new DO-178C and FAA/EASA memos updating DO-254 result in increased costs for those users taking shortcuts with the former DO-178B and early DO-254. However, these

guidelines possess attributes common to all safety-critical domains: planning, consistency, determinism, thorough documentation and testing, and proof of the preceding attributes. Truly, DO-XXX/ARP47XX rely upon significant verification to assess aviation product quality. However, true quality comes from a quality design, and implementation process, not from testing.

These aviation “standards” presume that aircraft, systems, hardware, and software must operate in harmonic unison, each with proven reliability. They provide an overall “Ecosystem” including Safety, Systems, and ancillary guidelines as depicted below:



The Basic Aviation Development & Certification Ecosystem

Before DO-254, hardware was considered “visible” and tested at the systems level with integrated software; hence hardware was exempt from DO-178B’s objectives. But that exemption resulted in functionality being moved from software to hardware for the purpose of avoiding software certification. Also, hardware complexity has evolved such that hardware is often as complex than software, due to the embedded logic within the PLDs, ASICs and FPGAs. Now, everyone recognizes that hardware and software comprise an inextricable chain with the quality equal to that of the weakest link, hence the mandate to also apply DO-254 to avionics hardware; the weak links are continually removed through the avionics certification evolution. ARP4754A requires explicit consideration of this entire system throughout the aviation

development lifecycle and DO-178C/254 thus align with ARP4754A.

In DO-178/278 “software” pertains to all drivers, BSP, OS/RTOS, libraries, graphics, and the application software - in other words, any executable logic or data that is loaded into memory or accessed during real-time execution. Testing means ensuring that the lowest level detailed requirements are accurately implemented, paths are covered according to their criticality level, and full bi-directional traceability is provided. Increasingly, tools are being used to automate DO-XXX and ARP47XX processes. For example, almost all compliant aviation projects today use a commercial traceability tool to manage links between requirements, implementation, and tests (see the Traceability chapter within this book). While such traceability tools are not formally required, virtually every compliant project uses them to meet DO-XXX/ARP47XX top-to-bottom and bottom-to-top traceability mandates. While there is no formal requirement for such tools (you can always provide traceability manually), a DO-XXX/ARP47XX compliant traceability tool greatly reduces the cost of compliance for all but the simplest of projects. Today, the same is true for configuration management tools, automated testing tools, structural coverage analysis tools, and logic static analysis tools: many such tools have come to market explicitly to meet compliance needs for DO-XXX and ARP47XX. As noted previously in this book, the acquisition cost and learning curve of these engineering automation tools mean the first project team to use them is likely not cost-effective; it’s the same for building reusable software and hardware components. However, subsequent projects greatly benefit from these tools and improved reusability, and that is where DO-XXX/ARP47XX shine.

Aviation Industry Avionics Costs.

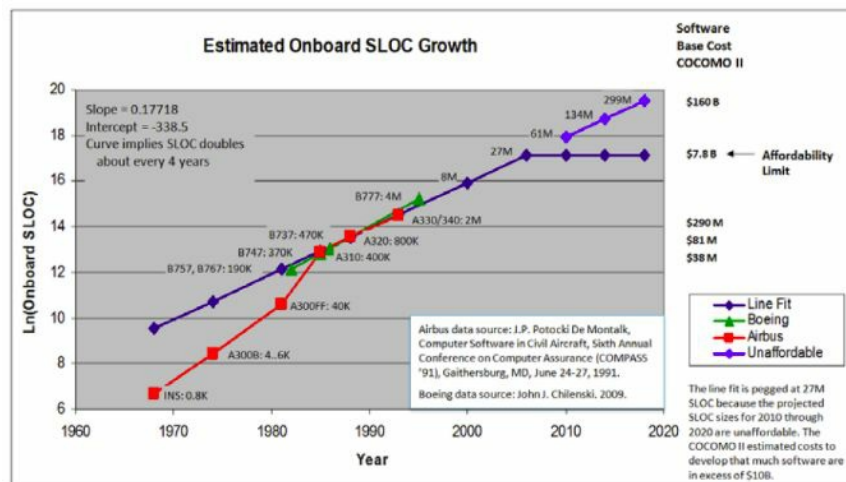
Safety-critical systems are measurably more expensive to develop than consumer devices and commercial Information Technology (IT) systems. Essentially, the improved fault tolerance and higher reliability come at a significant price. As aviation systems become more complex, the engineering R&D costs associated with that increased complexity likewise incur an exponential increase. The table below is extracted from a funded study performed by a large aviation development consultancy, with the data having a margin of error of 20% (due to inability to validate a portion of the

assumptions and data).

Aircraft Type	Approx Total Lines of Software & Logic	Average Cost per Line of Software & Logic	Cost of Software Development
Piston-Powered General Aviation, 6+ seat	100K – 1.5M	\$260	\$26M - \$390M
Gulfstream G550	4.0M	\$330	\$1.32B
Boeing 737 NG (600) (1995)	3.5 M	\$240	\$700M
Boeing 777 (legacy - 995)	5.5 M	\$260	\$1.04B
Boeing 787 (2011)	7M	\$370	\$2.59B
JSF-35 (Joint Strike Fighter, as of 2019)	15M	\$520	\$7.8B (onboard)

Aircraft Software Engineering Cost Synopsis per Aircraft Type

In a separate study, Carnegie Mellon University's Software Engineering Institute (SEI) compares and contrasts a related data set of Airbus and Boeing software Source Lines of Code (SLOC), depicted in the figure below. Note that SEI states that SLOC's will double every four years and they also assume a larger code base per aircraft type than the prior study in the preceding figure:



Source: Software Engineering Institute, Carnegie Mellon University

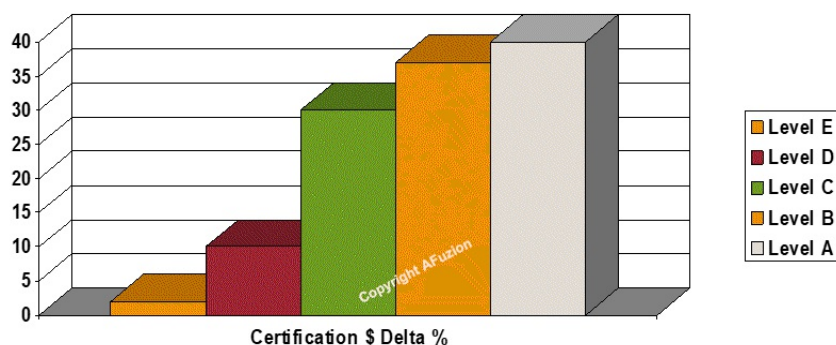
It is interesting to note that the Apollo space program had great success with their development yet had little in the way of development standards. However, their system complexity and logic size were less than 1% of today's commercial aircraft, with zero concern for certifiability or reusability. In fact, since aircraft today largely enhance previously developed systems,

the reusability of those systems is paramount: this is where DO-XXX/ARP47XX greatly contribute to increased reuse with reduced integration time as detailed below.

DO-XXX/ARP47XX Criticality Levels and Cost.

DO-178 and DO-254 entail five different levels of criticality, ranging from Level A (most critical) to Level E (least critical). DO-200 has three such levels and DO-278 has six; all for reasons explained within their corresponding chapters previously within this book. Each aviation system is assigned one or more levels of criticality via a safety assessment, and each item within that system must meet or exceed its assigned criticality level. As the criticality level increases, so does the degree of rigor associated with increased documentation, design, reviews, implementation, verification, quality/process assurance, and independence. Accordingly, cost and schedule increase as well. The following graph, and associated table below, depict the actual cost deltas associated with each level of criticality: as a baseline, the additional costs of DO-178C are contrasted with medium-quality software development, e.g. high quality commercial or consumer software (where the applicable true CMMI level of development was between Level 3 and 4).

Graph: Delta Cost and Schedule of DO-178 per Criticality Level, Above CMMI 3



Level E Cost	Level D Cost	Level C Cost	Level B Cost	Level A Cost
Baseline	E+15%	D+35%	C+10%	B+5%

Table: Delta Cost and Schedule of DO-178C per Criticality Level, Vs \$

A popular myth is that DO-XXX/ARP47XX are overly expensive and

increase costs by more than 200%. Indeed, they are not “cheap” as clearly the additional initial costs can be seen above. Level D certified systems still have generally full planning, good requirements with traceability to tests, documented implementation, reviews, and full functional testing of all requirements with traceability applied. Additionally, configuration management, quality assurance, and certification liaison are applied even to Level D. Remember, per the prior DO-178 chapter, that Level D has 26 Objectives, which is 26 more than your smartphone or amateur drone have. Yet the costs of Level D are just 15% higher than medium-quality consumer/commercial software developed per industry-typical CMMI Level 2-3 processes. Why? Because Level D is comprised almost entirely of normal industry-standard engineering principals; again, those are: plans, requirements, tests, traceability, reviews, and basic CM/QA. Also, many companies new to DO-XXX/ARP47XX believe their prior, existing efforts including planning, requirements, designs, test, and reviews must be re-done: not true. In fact, a capable Gap Analysis activity will analyze the existing processes and work to re-use such processes while identifying the “gaps” to fulfill DO-XXX/ARP47XX requirements.

DO-XXX/ARP47XX also require reviews (mostly independent for Levels A and B) of all software development processes and artifacts. Reviews must be per approved checklists to ensure all required aspects are properly reviewed. To save money and time, many companies use automated project management/review tools, vendors of which any internet browser will reveal (such a sample checklist is included in the Appendix of this book).

Another myth is that the most significant software cost escalation occurs when moving from Level B criticality to Level A. Untrue, but for an interesting reason. The singular largest difference between a Level A system and the Level B system is the 100X greater reliability required by the Level A system per ARP4761A. However, that 100x reliability must come primarily from the system/hardware architecture and not the software. How? Added redundancy: usually the only way to meet Level A reliability is via increased hardware/system redundancy which of course greatly increases total product cost. But the software cost difference between Level A and Level B software is quite minor as seen in the figures above (+5%).

The most significant cost differential within DO-178C is between Level D and Level C (same for CNS/ATM DO-278A Level 3 and 4). Why? Level C requires the following key objectives which Level D does not, and which results in Level C requiring 35% more effort than Level D:

- Testing of low-level software requirements
- Ensuring 100% coverage of all source code statements
- Assessment of requirements, design, and code to standards
- Greater rigor placed upon reviews
- In many cases more rigorous configuration management

Level B requires additional structural coverage (decision-condition, e.g. all branches in the source code), additional independence in reviews, and tighter configuration management. At first glance, it would seem that Level B should be significantly more expensive, e.g. 50% - 70%, than Level C. In theory, it seems to make sense, but as in many areas of life, common sense overcomes theory. In Level B (and C) there must be detailed, low-level software requirements and they must be thoroughly tested. Remember, DO-178C requires detailed low level requirement verification beginning with Level C and those low-level requirements will cover the vast majority of software logic decisions. During requirements-based testing, most (80-90%+) of the branches in the source code are already covered and hence require no additional structural coverage testing if test capture and coverage tools are appropriately used during that functional testing. Therefore, the seemingly significant cost increase associated with Level B versus C structural coverage is already mitigated by DO-178C's greatly enhanced requirement-based testing. Also, quality software engineering organizations already incorporate a semi-automated and streamlined process which includes independent reviews and tight configuration management; ergo the added cost of those aspects for Level B is largely mitigated.

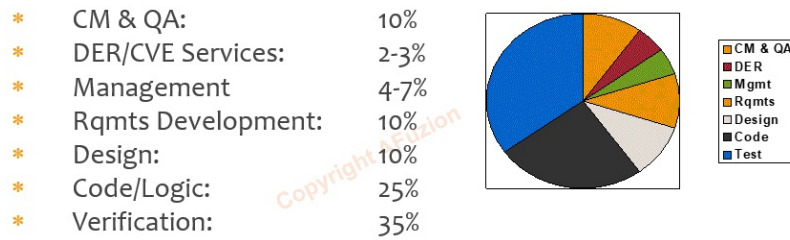
Now, "efficient" followers of the former DO-178B will find even greater cost impact in adhering to DO-178C. Their "efficiency" may have been due to taking liberties with DO-178B's intended, but less enforced, low-level requirement detail. Such shortcuts enabled less detailed functional testing

with many fewer logic branches verified. While acceptable for DO-178B Level C, it's unacceptable for DO-178C Level C which mandates greater details of low-level requirements. As a result of that greater detail, DO-178C inadvertently reduced the difference between Level C and Level B because the decision-condition structural coverage objective of Level B is largely covered already in Level C due to those more detailed low-level requirements. Also, developers making extensive use of Parameter Data Items (objects or logic external to the main application program) are now required to fully document, review, trace, and test all that data under DO-178C; something they "should" have been doing under the intent of DO-178B. Et Voilà.

Level A is the most critical software level and hence the most expensive. True. Still another myth exists for Level A logic, namely, "Level A is extremely difficult to achieve and the logic will cost at least 30-50% more than Level B." False. Level A logic imposes yet more structural coverage requirements (MCDC testing), source to binary correlation, and more independence within reviews. The most significant cost driver over Level B is the MCDC testing requirement. However, with proper application of modern structural coverage tools, personnel training, and thorough requirements based testing, the added cost for Level A can be largely contained, thus Level A software is only slightly more expensive than Level B. For this reason, most COTS products pursuing Level B certifiability instead opt for Level A. However, as previously mentioned the system/hardware costs will be higher for Level A than B due to added redundancy required to meet Level A's 100x higher required reliability over Level B (10^9 vs. 10^7).

Where does the money go?

The following chart shows a typical breakdown of expenditures for a Level (DAL) B project; Level A of course would have a greater percentage allocated to Verification. This chart is for conventional structured, not model-based, design (MBD would have much larger requirements/design allocation with greatly reduced coding time presuming autocode generation from the models was used).



- Primary Cost Drivers?
1. DAL assignment/mitigation per FHA
 2. Accurate & Detailed Requirements
 3. Accurate & Thorough Reviews to Prevent Defects
 4. Minimizing Code/Logic Changes
 5. Efficient Testing : less overlap, more automation

Certification Cost Metrics (DAL B with Structured Design)

As shown in the above figure, the top five cost drivers are all supported by efficient implementation of DO-XXX and ARP47XX.

Benefits of DO-XXX/ARP47XX.

Aviation's guidelines are certainly neither free nor even cheap, as cited above. However, they can actually be cost-effective, when implemented properly. Particularly when evaluated over a product lifetime or subsequent new product versions when efficiency and benefits are most notable. A summary of DO-XXX/ARP47XX benefits is provided in the following figure, with details following:

1. Earlier Defect Prevention & Fewer In-Field Defects
2. Greater Reusability
3. Improved Visibility & Confidence of Change Impact
4. Improved Visibility by Management, and of Suppliers
5. Quantified & Improved Safety Determination
6. Reduced Cost, Schedule, & Risks
7. Fewer Assumptions = Fewer Defects
8. More Known & Complete Test Coverage
9. More Complete Documentation ➔ Easier Maintenance
10. Wider & Deeper Market Penetration
11. Higher Quality

Essential DO-XXX & ARP47XX Benefits

Efficient and experienced aviation development organizations achieve the above benefits of DO-XXX/ARP47XX, further clarified below.

- Greater upfront requirements clarity: DO-XXX/ARP47XX mandate thorough and detailed requirements. Such detail, and the necessary discipline, force answers to be provided up-front instead of being deferred. Assumptions are drastically minimized. Consistency of requirements and their testability is greatly enhanced. Iterations and rework due to faulty and missing requirements are greatly reduced. Yes, other standard and guidelines such as CMMI mandate such upfront requirements; however, DO-XXX is unique in its enforcement of requirement detail.
- Fewer implementation iterations: Implementation iterations, or churn, are the bane of aviation engineering. For example, in many cases, ten, twenty, and even thirty versions of evolving code files exist on new products. Incredible. With strong engineering processes and discipline, implementations should be largely correct the first time they are developed and should not require dozens of updates to get it right. Models and implementation should be reviewed by analyzing implementation versus documented requirements.
- Fewer bugs found during developer testing: Since DO-XXX and ARP4754A mandate thorough and testable requirements, in addition to code reviews for Level C software and above, far fewer defects will be found during developer testing. Independent logic reviews required for Level B and A go beyond this goal. And logic reviews require the following completed and configured items prior to writing models or code:
 - Standards & checklists
 - High and low level requirements
 - Design
 - Traceability for the above
- Greater consistency within software: aviation systems are like a chain: they can only be as strong as their weakest link. A system which is 99% correct is 1% incorrect, which means it is unsafe. Systems are never provably perfect and DO-XXX/ARP47XX make no claims that perfect

adherence to their objectives yields perfect systems. The weakest logic module, or engineer, is on the critical path of aviation safety. All systems must be consistent per their level of criticality and DO-XXX/ARP47XX properly implemented enforce such.

- Fewer defects found during integration: Integration can be a lengthy iterative process where major defects requiring design changes are discovered and fixed. Not with DO-178 and DO-278, where integration is typically 50-75% faster than non-DO-XXX environments. Since all software whose operation is potentially affected by hardware must be tested on actual target hardware for the higher DAL's, integration begins earlier, and software is verified earlier. Also, the system upon which the software resides must comply with ARP4754A, which mandates similarly strong system processes; it's likely the custom hardware logic components will need to meet DO-254 (hardware logic's corollary to DO-178C) so that hardware will similarly be of higher quality at integration. Thus, integration begins earlier with inherently fewer flaws than non-DO178C environments.
- Improved configuration management: The ability to control and recreate any snapshot of the project is covered by Configuration Management (CM). However, CM also ensures security, backups, completed reviews, problem reporting, and version control. DO-XXX and ARP47XX have strong CM requirements, coupled with modern tools which automate many CM tasks, ensure higher and more provable product quality now and in the future. Literally any version of certified systems must be able to be recreated and retested in its entirety for thirty years after delivery. That means not only the certified implementation must be controlled and captured, but all the ancillary files including makefiles, build scripts, 3rd party tools, development environments, design and test data, etc. must all be captured. This capture eliminates common problems in system maintenance found within other industries.
- Easier regression testing: You build your product on the assumption that it will be successful, and therefore have a long life. You also know your software will evolve through new applications, installations, and versions. Regression testing can be expensive if extensive or manual;

DO-XXX and ARP4754A provide thorough traceability to determine which modules need changes or analysis, and corresponding retesting. Since aviation requires thorough and provable regression testing, DO-XXX/ARP47XX adherents receive the benefit of easier and usually automated system, software, and hardware retesting.

- More thorough testing: 100% requirements coverage, greater robustness coverage with clarity, and logic structural coverage are deterministic verification aspects of DO-XXX. Parameter Data Items must be particularly well documented and verified under DO-178C. Unused software structures (dead code) must be removed, and deactivated code documented and verified for Levels C through A. The probability of software errors encountered in the field is vastly reduced, particularly at Levels A and B.
- Improved hardware and system testing: System testing is typically challenging for embedded systems because the development environment is quite removed from the target environment. Also, testing is often performed via different engineering teams. The improved determinism and quality of all these components via DO-XXX belies improved system integration. Since DO-XXX and ARP47XX require complete traceability, it is quite possible to show via traceability that system requirements were actually fully tested during software testing and therefore need not be repeated by additional testing at the system level.
- Fewer in-the-field defects: In-the-field defects and customer returns are hugely expensive in both the short- and long-term. Products developed under DO-XXX have been demonstrated to result in 80%-90% fewer returns.
- Improved reusability: Via the thorough and consistent documentation required by DO-XXX, modularization, enforcement of documented modern engineering principles, and reviews to ensure all the above was achieved, reusability is greatly improved. In aviation, reusability is the Holy Grail, but the reality is that unless a component is at least 80% reusable (i.e. unchanged), then it is often quicker and less risky to simply start over from the beginning. In reality, most software is less

than 50% “reusable”. With DO-XXX, and enforcement of design/coding standards, coupled with independent reviews and traceability, most modules should be at least 70% re-usable. And when modeling and C++ are employed, reusability increases even more.

- Improved customer satisfaction: Reliable systems beget reliably returning customers. Decreased single-point failures are a direct result of DO-XXX. Software is an art; artists resist documenting their work and subjecting it to common development standards and peer reviews. Without standards, discipline, and modern software engineering principals, software teams devolve into a group of loosely structured rogue artists; these artists are highly valuable, creative, and talented persons. However, the loss, or deficiency, of any such artist for any reason is catastrophic to the team, unless their work is documented, understood, and consistently applied as for the other artists. DO-XXX greatly reduces the possibility of such single-point personnel failures.
- Improved management awareness of true schedule status: How many aviation projects report a “99% Complete” status week after week? How is project progress measured? How can management truly ascertain completion status of nearly invisible logic? The answer to all these questions is via modern and accurately detailed management techniques built around DO-XXX. The provision for insight, traceability, accurate status on design, development, testing, integration, and reviews is found in the aviation guidelines.
- Improved completion schedule due to the above: “What gets measured gets done.” DO-XXX/ARP47XX provide for continuous project completion and quality status-ing.
- Wider market acceptance: The aviation guidelines provide for a widely known proof of reliability. Safety critical applications abound in aerospace, medical, transportation, energy, and even entertainment (think amusement parks); all these fields have lives at stake and are in need of improved safety, reliably so.
- Improved estimation of costs associated with changes and revisions. Since DO-XXX includes full visibility of logic internals, traceability of

requirements, design, code, and tests, along with metrics kept by Quality Assurance, this enables clearer determination of the impact associated with logic changes.

- Improved differentiation vis-à-vis competitors: Capitalism ensures that competitors compete, which means they will continuously adopt a competing product's best features thereby eroding differentiation hence market share. The answer? Evolve your own products and differentiate them yourself. Companies, and products, that do not grow are like trees that do not grow. They soon die off. Aviation systems are dramatically growing in complexity while development training and engineering barely keeps apace. Many companies understand that proving your reliability, your quality, and your value-added proposition by adopting DO-XXX/ARP47XX. Far from a rubber stamp, the aviation guidelines require proof. "Guilty until proven innocent" is the mantra of aviation reliability and the guidelines.

Improved contractor/subcontractor/supplier work-sharing quality: DO-XXX/ARP47XX process document rigor and completeness help avoiding misunderstandings and errors between companies and contractors/subcontractors/suppliers, fostering an effective work sharing among organizations.

UAV & UAS Certification Aspects

By Greg Wilson, FAA DER

Reviewed by:

- Mr. Vance Hilderman, CTO AFuzion Inc.

Over the last 100 years, manned aircraft and systems have experienced dramatic growth resulting in a significant increase in both aircraft cockpit automation and advances in avionics system designs complexities. This automation of the cockpit includes advances ranging from fly-by-wire, auto-flight and aircraft automation. Aircraft and technology advances have reached a point where flight computers are often used (or required) to augment or control modern aircraft.

As these manned aircraft systems have increased in complexity and sophistication, and the importance and role of these systems within the aircraft has expanded, the potential to autonomously operate aircraft in mixed airspace has taken center stage! This progression towards autonomous aircraft has resulted in the evolution of aircraft designs to include Unmanned Aircraft. For several years Unmanned Aircraft Systems (UAS) have been under development, primarily operated by world military organizations. Now UAS are poised to become an integral part of the airspace that was previously allocated to manned aircraft operations. In the United States and throughout the world the push is on to include UAS in the air space currently restricted to manned aircraft. The interest in commercialization of UAS has skyrocketed over the last few years and the desire by commercial entities to operate UAS in the United States national airspace (NAS) as well as many other countries, has reached a fever pitch. The commercial use of UAS includes numerous applications ranging from aerial photography, to search and rescue, reconnaissance, inspection such as utility line inspections, event monitoring, reporting, fire spotting / fighting, cargo delivery, infrastructure monitoring, communications and broadcast, disaster response, farming, etc. These are just a few examples of the commercial use of UAS. As UAS are integrated into the NAS many new applications that have not even been thought of as yet

will emerge!

Background & History.

When one considers the overall timeline manned aircraft rules and regulations have required, changing these rules and regulations to accommodate UAS operations in the NAS in a relatively short period of time represents a significant challenge. The development of manned aircraft rules and regulations has consistently focused on safety. Airworthiness standards for existing US aircraft are defined in the US government code of federal regulations (CFR), Title 14 CFR, and processes for FAA type certification are defined in FAA Order 8110.4 and airworthiness certification in FAA Order 8130.2. As one would expect, changing these rules and regulations to include UAS in the NAS was, and still is, a very complex process requiring a significant amount of time and resources. Currently the FAA and EASA are still establishing the rules and regulations regarding UAS operations, though much progress has made in this endeavor over the past few years. Any entities wishing to design, manufacture, market, or operate a UAS larger than minimum weight thresholds in the NAS must obtain FAA approval for the UAS: the UAV, UAV pilot, and UAV operational environment/plans. For type design approval (see the Chapter in this book on Type Certificates, Technical Standard Orders, and Parts Manufacturing Approvals for further explanations), UAS designers must show they meet acceptable safety levels for the UAS design, and the UAS operators must employ FAA certified systems that enable compliance with standardized air traffic operations including contingency/emergency procedures for UAS. As part of the FAA UAS work, the FAA is harmonizing with EASA and the international community for the completion of civil aviation and UAS operations.

Because the primary focus of the FAA is safety, ultimately UAS operations in the NAS will necessitate UAS satisfying FAA safety of operations requirements while conducting flight in the NAS. Since the early 1990's UAS have been allowed to operate in the NAS on a limited basis. In the beginning, these UAS operations were limited to military and law enforcement. FAA Notice 8900.207 was issued in January of 2013 and was later cancelled and replaced with N 8900.227 Unmanned Aircraft Systems (UAS) Operational Approval. This notice provided information regarding the

evaluation of safety of UAS, interoperability of UAS, and UAS flight operations in the NAS and can be used for the development of COAs allowing access to the NAS. The special airworthiness certificate falls into the experimental aircraft category and requires specific proven capabilities to enable operations of aircraft in the NAS. Civil UAS were then accommodated with experimental certificates under FAA Order 8130.34. A special experimental airworthiness certificate also includes constraints the aircraft operational usage. The FAA reviews each UAS experimental airworthiness application on a case-by-case basis. This case-by-case review enables the FAA to carefully define a level of access that is limited and dependent on risk mitigations for the UAS and operations of the UAS. These limitations ensure safety and efficient UAS use of the NAS and ensures that use of the NAS is not diminished. The use of a special experimental airworthiness certificate for UAS is similar to the manned aircraft experimental airworthiness certificate and is normally issued to UAS applicants for the purposes of research and development, crew training or market surveys per 14 CFR 21.191(a), (c), and (f).

The FAA developed a notice of public rule making (NPRM) to allow Small UAS (sUAS) to conduct operations in the NAS. This NPRM action included FAA stakeholders who have an interest in sUAS operations in an effort to develop a consensus standard(s) required for rule making and rule implementation. The FAA then worked with the International Civil Aviation Organization (ICAO) (a special agency of the United Nations) that promotes safe and orderly development of international civil aviation throughout the world. ICAO sets standards and regulations for aviation safety and security. ICAO addressed UAS to provide a fundamental international regulatory framework to support UAS operations throughout the world. ICAO has issued guidance requiring Member States to implement Safety Management System (SMS) programs. These programs are essential to manage the risk associated with use of UAS in the aviation system. UAS operations introduce numerous new challenges, including the ability to analyze massive amounts of data to provide useful information for oversight and assessment of risk. ICAO Circular 328 provided guidance regarding introduction and integration of UAS into the airspace. The objective, as defined in this ICAO circular, is to provide guidance to ensure interoperability and regulatory compatibility,

when possible. The FAA then worked with committees such as RTCA (Special Committee SC-203 and SC-228) to define the Minimum Performance Standards (MPS) and Minimum Operational Performance Standards (MOPS) for UAS operations in the US NAS. The FAA then published a document outlining a roadmap and plans for integrating UAS into the NAS. The title of the FAA UAS road map was “Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap” First Edition – 2013 (published November 2013). This FAA UAS roadmap was developed by the FAA and the UAS Aviation Rulemaking Committee (ARC) to outline the FAA’s plans and objectives for integrating UAS in the NAS. The FAA roadmap outlined the general timelines, tasks, and considerations needed to integrate UAS operations in the NAS. Then, the committees progressed on development of UAS Minimum Aviation System Performance Standards (MASPS) and development of Minimum Performance Standards (MOPS) used in support of Technical Standard Orders (TSO) (mid-term to long-term time frame).

Like manned aircraft and manned aircraft systems development processes, most UAS follow a development process much like that of manned aircraft and manned aircraft systems. One of the differences will be in the assessment of the UAS safety case. Before one can begin to develop the requirements for a UAS, a comprehensive safety case needs to be developed for the UAS operations, and this safety case then becomes the basis for the definition of the system safety assessment including software (DO-178C) and hardware DO-254 (airborne electronic hardware) development assurance level(s). Like manned aircraft and systems, the UAS system safety assessment will rely upon SAE ARP4754A Guidelines for Development of Civil Aircraft and Systems and SAE ARP 4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. Additional guidance material that may be useful for the development of the safety case is contained in NATO STANG 4671 Unmanned Aerial Vehicle Systems Airworthiness Requirements (USAR) and STANAG 4586 Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability. If one were to follow manned aircraft objectives regarding the development of the safety case/safety assessment and directly apply this to the UAS safety case/safety assessment, this potentially could result in a UAS that is

developed to a level of safety beyond what may be required. On the other hand, if the UAS safety objectives are set too low, the UAS development may not include a sufficient level of safety and therefore may be at risk of not meeting the FAA UAS requirements and regulations (UAS MPS/MOPS).

Along with the typical challenges of developing any system, the UAS safety case must address the followings:

- UAS Pilot and Crew:
 - Training
 - Operational standards
 - Certification requirements
 - Operational procedures
 - Regulatory requirements
 - Medical Standards
 - Testing Standards
- Control Station:
 - Certification Requirements and basis of certification, for instance (DO-178C and / or DO-278A Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance)
 - Technical Standard Orders (TSO) as applicable
 - Airworthiness Standards
 - Regulations
 - Interoperability Requirements
 - Guidance Materials (Advisory Circulars, Orders, Notices, etc)
 - Continued Airworthiness
 - Means of Compliance (MOC)
 - Voice command and control (communications with FAA Air Traffic Control – ATC)
- Data Link:
 - Certification Requirements and basis of certification, for instance (DO-178C and / or DO-278A Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance)
 - Technical Standard Orders (TSO) as applicable

- Airworthiness Standards
- Interoperability Requirements (as applicable)
- Guidance Materials (Advisory Circulars, Orders, Notices, etc.)
- Coordination of Aviation Radiofrequency Spectrums and use
- Standardized Control Architectures
- Measure of Performance
- Radio / Datalink Security Requirements
- Means of Compliance (MOC)
- Regulations
- Unmanned Aircraft:
 - Certification Requirements and Certification Basis
 - Technical Standard Orders (TSO)
 - Airworthiness Standards
 - Procedures
 - Regulations
 - Guidance Materials (Advisory Circulars, Orders, Notices, etc)
 - Measure of Performance
 - Continued Airworthiness
 - Testing Standards and Records
 - Means of Compliance (MOC)

Additional information and guidance are available from DO-304 Guidance Material and Considerations for Unmanned Aircraft Systems, DO-320 Operational Services and Environmental Definition (OSED) for Unmanned Aircraft System, and DO-344 Vol 1 & 2 Operational and Functional Requirements and Safety Objectives for Unmanned Aircraft Systems Standards. FAA regulations are based on manned aircraft rules and regulations; historically these rules and regulations required the Pilot In Command (PIC) to “See and Avoid” requiring the PIC to maintain constant situational awareness. Introduction of UAS requires an evolution to “Sense and Avoid” instead of “See and Avoid”. In manned aircraft, See and Avoid, coupled with radar, visual sighting, separation standards, and proven technologies have ensured safe operation of manned aircraft. With UAS a paradigm shift is required so as to transition from “See and Avoid” to “Sense and Avoid”. This paradigm shift places significant emphasis on the UAS sensors suite (both ground and airborne) since the sensor suite will replace

the manned aircraft PIC's function to maintain situational awareness. Additionally, the role of control and communication (C2) also requires a paradigm shift since UAS rely on a ground based C2 link.

UAS C2 is evolving, with significant research and development progressing to assess the viability of C2 for UAS. The UAS C2 link between the unmanned aircraft and the ground control station requires the definition of the C2 requirements and C2 required performance. Currently a substantial effort is underway to assess UAS control data link communications, including assessment of latency and latency issues, integrity, continuity, availability, jamming, functional failures and / or backup systems, and numerous other performance criteria. This results in:

- Definition of the UAS C2 requirements
- Definition of the UAS C2 performance criteria
- Assessment and definition of UAS C2 link vulnerability and security
- Assessment and definition of the UAS C2 communication spectrum (this requires coordination with US and international radio communications entities)
- Definition of means of mitigation
- Establishment of end-to-end UAS performance measurement criteria
- Development of Next Generation concepts for UAS 4-dimensional trajectory criteria

In support of this UAS C2 effort multi R&D projects are focused on the development and validation of UAS control link systems, quantification of the UAS safety critical communications systems, and simulations of UAS C2 performance. The development of UAS C2 is a collaborative effort involving the FAA, US government agencies, industry, and international stakeholders.

UAS SAA is also evolving, with significant research and development underway to assess the viability of SAA for UAS. As part of this R&D effort sensor suites and types of sensors (both airborne and ground-based) are being assessed. A primary part of the UAS SAA R&D effort is focused on the assessment of multi-sensor use and integration of various sensors to produce

a composite sensor suite that is capable of satisfying the requirements for UAS SAA. For the most part, SAA falls into one of two categories: Ground Based SAA and Airborne SAA (UAV sensor suite). The development of UAS SAA is a very complex and difficult task that present unique and challenging issues. Complicating the development of UAS SAA is a lack of requirements. To close the gap created by a lack of UAS SAA requirements, the FAA researched UAS SAA. The FAA research included the following tasks:

- Define and establish UAS SAA system definitions and performance goals;
- Assess various UAS SAA implementations and UAS SAA multi-sensor systems, and use of multiple technologies;
- Define and establish the minimum set of information required for UAS collision avoidance maneuvering.

Research including actual test flights of ground based UAS SAA and airborne UAS SAA is underway. Because of the complexities of airborne UAS SAA the expectation is this technology may not fully emerge for several years, thus ground based UAS SAA may emerge as the initial form of UAS SAA.

One additional topic that should be discussed centers around UAS and human factors (HF). The inherent paradigm shift that created as UAS enter the airspace previously assigned to manned aircraft will result in new and challenging issues pertaining to UAS ground-based pilot communications with the NAS Air Traffic Control (ATC) and Ground Control. This required the collection and analysis of UAS pilot interactions with ATC during UAS operations in the air space populated with other users. This obviously is a challenging topic since one would expect that the collection and analysis of the UAS pilot interactions with NAS ATC would require actual UAS operations in the NAS alongside other civil airspace users. The US Government Joint Planning and Development Office (JPDO), in collaboration with academia, industry, and research facilities, identified several challenges regarding UAS HF including:

- HF pertaining to performance requirements (UAS, ATC, etc.);

- HF pertaining to the qualifications and criteria / requirement to become a UAS pilot;
- HF pertaining to the identification and definition of communications between the UAS pilot, UAS ground controller, ATC, dispatcher, etc.;
- HF pertaining to UAS operations resulting from: lost link, lost communications, workloads (both UAS pilot / ground controller and ATC);
- HF pertaining ATC aircraft separation and display of this information;
- HF pertaining to the ground control station (GCS), GCS displays, GCS use and operations;
- HF pertaining to UAS situational awareness.

As was mentioned earlier, the assessment and analysis of UAS HF requires collection and assessment of actual UAS flights in the NAS alongside other civil airspace users. Since UAS operations in the NAS are predicated on the analysis of UAS pilot interactions with ATC use of some limited military UAS flights (restricted airspace) and other data that could be developed through use of the planned FAA UAS test ranges may satisfy the need for actual UAS pilot interaction data. The minimum set of data that will need to be collected includes:

- Collection of UAS pilot operations;
- Analysis of ATC will provide service to airspace users (both manned aircraft and unmanned aircraft);
- Analysis of ATC controllers' interactions with the UAS pilot;
- Analysis of automation and uses of automation.

Along with the UAS Pilot and UAS systems, the FAA also developed revisions to the Air Traffic Operations (ATO). UAS operations present unique and challenging conditions that manned aircraft operations have not previously needed to be addressed. Therefore numerous Air Traffic policies and procedures will need to be revised, re-written, or in some case created from scratch. The modification of these Air Traffic processes, and procedures will in part be based on research, analysis, and simulations of UAS

operations in the NAS. The following identifies key ATO processes/procedures that were necessary to consider changing included:

- Controller:
 - Policy
 - Handbooks
 - Training
 - En Route, Terminal and Oceanic operations
- Operations:
 - Policy
 - ATC Management
 - Flight Planning
 - Separation and Flow Control
 - Normal Procedures
 - Contingency Procedures
 - Return to home
 - Loss of control
 - Lost Communications / Lost Telemetry
 - En Route, Terminal, and Oceanic Procedures
 - UAS Airport Surface Integration
- Safety:
 - Data Collection and Analysis
 - Safety Case
 - Safety Requirements
 - Post Implementation Assessment
 - En Route, Terminal, and Oceanic Procedures

Further complicating UAS integration into the NAS is the wide range of UAS that are expected to request access to the NAS. UAS designs range from very small to very large with widely varying performance capabilities and requirements. Currently regulatory requirements for manned aircraft voice communications responses to ATC instructions do not include a quantitative response time. Since the current regulations did not include voice communications response times, for UAS this potentially required communications response times to be an inherent part of the UAS system design criteria. UAS operations in the NAS interjected some amount of

operational latency between the UAS and the exchange of voice communication between the UAS controller and ATC. Also, Control and Communications (C2) requirements that are currently being defined will include the use of third party communications services that are currently and integral part of manned aircraft operations.

In Summary, as can be seen the integration of UAS into the NAS was, and still is, an extremely complex endeavor. Multiple issues needed to be addressed before UAS were allowed to fly in the NAS and these are still evolving. The FAA's near-term, mid-term, and long-term focus for UAS operation in the NAS is focused on safety. Near-term access to the NAS for UAS is based on enhanced procedures and advancements in technology, which accelerated UAS access to the NAS. For the near-term, R&D on advanced mitigations helped to facilitate FAA accommodation of UAS into the NAS. This R&D, along with appropriate restrictions and constraints, is intended to mitigate any UAS performance shortcomings and facilitate near-term integration of UAS in the NAS.

Conclusion

“It was a dark and stormy night ... that eventually gave rise to sunshine.”

All things come to an end, if such is only transitory. Like an ever-evolving ecosystem, the forest is never completed. Rather, the wise forester understands enough to be able to generally predict and guide future changes. Similarly, no book on complex aviation systems can ever be truly finished as such belies the dynamic nature of the technological world's ecosystem. Instead, you now have the basic tools to understand that continuum and help guide aviation's future toward a better harvest.

Some readers undoubtedly wanted a book that answered all their questions. Really. Irrespective of the fact that the number of potential questions is infinite whereas the pages of this book are limited, the unbridled optimism associated with such wondrous hopes is truly applauded. Ahhh, reality check: since the time of Socrates, good teachers have known they are most effective when laying foundations, guiding and encouraging the process of true learning which is only tangentially related to normal teaching. If you want to quickly satisfy a physical hunger, you need simply obtain the results of a cook. When you want to ensure prevention of life-time hunger, your need is different: you must learn the fundamentals of cooking and with skill and resources you'll be forever satisfied. This book thus provides a basic foundation of aviation development knowledge and may be used to help guide your search for additional answers so that you can participate more productively in your own technical nourishment. While perhaps a seemingly great book could provide answers to many of the reader's questions via thousands of pages, a good book provides the reader with the means to answer those questions themselves. Hopefully this is a good book, as I'm certain it's not such a “great” book as engineers simply do not make the time to read thousands of pages.

The world of high reliability and safety-critical systems is vast and growing exponentially. It would be an easy matter to produce volume after volume on this topic even if constrained to basic avionics development and certification issues. The result would soon resemble an encyclopedia collection which

beautifully collects shelf dust while the non-reading audience reinvents its truths. The brilliant Mark Twain is said to have once penned an uncharacteristically lengthy personal letter and introduced it with an apology: “I’m sorry, I have not the time to make it shorter.” The same is true of this book, though extreme efforts were made to keep the material herein concise, relevant, helpful, and above all, short.

I hope you truly enjoyed and benefited from reading this book. Remember, it’s a small world, so all the more reason to live large.

Safe skies,

Vance Hilderman

Sayings from Vance Hilderman, from 1989 when he started his first company through today after his seventh. As recounted from emails and interviews with a few dozen of his prior 500+ employees.

- *“A system may only be as safe as you can prove it to be ... but normally less so.”*
- *“If you can conceive, design, and implement in less than a month, it’s probably not a complex system.”*
- *“Software is aptly named: without proper containment, when you squeeze it you’ll wear it.”*
- *“Hardware is aptly named: once broken, it’s hard to fix.”*
- *“Quality is a mindset, not an add-on component.”*
- *“Software can resemble a zoo: good programming beasts require months to create a seemingly acceptable system, but the monkey can crash it in a moment.”*
- *“There is a simple difference between Simple and Complex: Simple requires expertise, while Complex requires faithful expertise.”*
- *“Companies more interested in revenue than long-term business success will eventually achieve neither.”*

- *“Positive actions MAY produce positive results; negative actions WILL produce negative results.”*
- *“Hire engineers by considering both experience and prior mistakes: the former will minimize the latter.”*
- *“Learning to fix car brakes is great fun – so should first be attempted on your friend’s car.”*
- *“Actions speak louder than words, and facts speak louder than actions.”*
- *“Basic math is a requisite skill for success – any entrepreneur can easily calculate a half-day of work as being twelve hours.”*
- *“Always have the aviation job applicant bring a one-page hardcopy resume: its most important use is in seeing how quickly the applicant can craft a paper airplane.”*
- *“When an apprentice lion-tamer tells the master-tamer ‘I am not afraid’, but brings his gun and refuses to lead the way, there is only one meaning: he is cautiously afraid.”*
- *“Hire the million-dollar candidate: the one whose last mistake cost their former employer that sum as that mistake is assured to be a one-time event.”*
- *“When big government shuts down for a day and the stock markets go up that day, the clear value of big government is revealed.”*
- *“Forgive, but not forget, the past: living for tomorrow will benefit from yesterday’s lessons”*
- *“If a picture is worth a thousand words, a good explanation is worth a thousand pictures.”*
- *“Focus the moment: when drinking wine, reminisce not the beer ...”*
- *“It’s a small world; all the more reason to live large.”*

Appendix A:

Avionics Acronyms

Avionics has its own language. Years ago my wife Colleen accompanied me to an aviation symposium where I was speaking and later told me “your aviation engineers do not speak English ... well, not the kind of English normal people understand”. Indeed. When listening to avionics engineers converse, or reading avionics development literature, it can be all too easy to be mystified by this “language” of “Avi-lish”, or “Aviation English”. But like any language, basic vocabulary with a little context easily demystifies. In my prior books and papers, I neglected to fully summarize the many acronyms such writings are replete with; many readers have asked me to prepare a simple acronym list. It was always on my To-Do list, but right below cleaning my car engine. So here are the most popular aviation development acronyms; and now, to clean that car engine ...

Safe skies,

Vance Hilderman

Vance Hilderman

vance.hilderman@afuzion.com

Acronym	Definition
A/D	Analog To Digital
A/P	Autopilot
ABAS	Aircraft-Based Augmentation System
ABROAD	ADS Broadcast
AC	Advisory Circular
ACAS	Airborne Collision Avoidance System
ACARS	Aircraft Communications Addressing And Reporting System
ACC	Area Control Center
ACE	Actuator Control Electronics
ACMS	Aircraft Condition Monitoring System
ACO	Aircraft Certification Office (typically FAA, e.g. USA)
ACP	Audio Control Panel

ACS	Audio Control System
AD	Airworthiness Directive
ADAHRS	Air Data And Attitude Heading Reference System
ADC	Air Data Computer, or Analog-To-Digital Converter
ADF	Automatic Direction Finder
ADI	Attitude Director Indicator
ADIRS	Air Data Inertial Reference System
ADIRU	Air Data Inertial Reference Unit
ADM	Air Data Module
ADR	Airborne Data Reception
ADS	Automatic Dependent Surveillance
ADS-A	Automatic Dependent Surveillance-Address
ADS-B	Automatic Dependent Surveillance-Broadcast
ADS-C	Automatic Dependent Surveillance-Contract
AEH	Airborne Electronic Hardware
AESA	Active Electronically Scanned Array
AFCS	Automatic Flight Control System
AFD	Autopilot Flight Director
AFDC	Autopilot Flight Director Computer
AFDS	Autopilot Flight Director System
AFDX	Avionics Full-Duplex Switched Ethernet
AGACS	Automatic Ground
AGDL	Air-Ground Data Link
AHC	Attitude Heading Control
AHRS	Attitude Heading Reference System
AIDS	Aircraft Integrated Data
AIM	Aeronautical Information Manual
AIP	Aeronautical Information Publication
AIR	Aerospace Information Report
AIS	Airmen's Information System
AL	Assurance Level
ALT	Altitude
	Acceptable Means of Compliance (sometimes called "Alternative Means of

AMC	Compliance”)
AMS	Air Management System
ANC	Active Noise Cancellation
ANN	Annunciator
ANR	Active Noise Reduction
ANSI	American National Standards Institute
ANT	Antenna
AOC	Aeronautical Operational Control
AOP	Airport Operating Plan
AOPA	Aircraft Owners And Pilots Association
APARS	Automatic Pressure Altitude Reporting System
APS	Auto Pilot System
APU	Auxiliary Power Unit
APV	Approach With Vertical Guidance
ARINC	Aeronautical Radio, Inc.
ARP	Aerospace Recommended Practice (via Society of Automotive & Aerospace Engineers)
ASA	Aircraft Safety Assessment
ASD	Aircraft Situation Display
ASDL	Aeronautical Satellite Data Link
ASIC	Application Specific Integrated Circuit
ASR	Airport Surveillance Radar
ASTERIX	All Purpose Structured Eurocontrol Surveillance Information Exchange
ASU	Avionics Switching Unit
AT	Auto Throttle
ATC	Amended Type Certificate
ATC	Air Traffic Control
ATCRBS	Air Traffic Control Radar Beacon System
ATCSS	Air Traffic Control Signaling System
ATCT	Airport Traffic Control Tower
ATIS	Automated Terminal Information Service
ATM	Air Traffic Management
ATS	Air Traffic Service

ATSAW	Airborne Traffic Situational Awareness
ATSU	Air Traffic Services Unit
ATT	Attitude
Avionics	Aviation Electronics
AVN	Office Of Aviation System Standards
AWACS	Advanced Warning And Control Systems
AWOS	Automated Weather Observation System
B RNAV	Basic Area Navigation
BARO	Barometric Indication, Setting Or Pressure
BDI	Bearing Distance Indicator
BGAN	Broadcast Global Area Network
BIT	Built-In Test
BITE	Built-In Test Equipment
BOM	Bill of Materials
CAA	Civil Aeronautics Administration
CAS	Calibrated Airspeed
CAST	Certification Authorities Software Team
CAT	Catastrophic (when referring to a “Safety” aspect; otherwise may refer to “Category”)
CAT I	Operational Performance Category 1
CAT II	Operational Performance Category II
CAT IIIa	Operational Performance Category IIIa
CAT IIIb	Operational Performance Category IIIb
CAT IIIc	Operational Performance Category IIIc
CC	Control Category
CC	Change Control
CCA	Common Cause Analysis or Circuit Card Assembly
CDA	Continuous Descent Approach
CDI	Course Deviation Indicator
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CDTI	Cockpit Display Of Traffic Information
CDU	Control Display Unit

CEH	Complex Electronic Hardware
CFIT	Controlled Flight Into Terrain
CFR	Code of Federal Regulations
CI	Configuration Item
CIDS	Cabin Intercommunication Data System
CM	Configuration Management
CMA	Common Mode Analysis
CMP	Configuration Management Plan
CMR	Certification Maintenance Requirement
CNS	Communication, Navigation, Surveillance
CODEC	Coder/Decoder
COMM	Communications Receiver
COTS	Commercial Off-The-Shelf
CPDLC	Controller
CPS	Cycles Per Second
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check or Cyclic Redundancy Code
CRT	Cathode Ray Tube
CS	Certification Specification
CSCI	Computer Software Configuration Item
CTAF	Common Traffic Advisory Frequency
CV/DFDR	Cockpit Voice And Digital Flight Data Recorder
CVE	Certification Verification Engineer
CVR	Cockpit Voice Recorder
CWS	Control Wheel Steering
DA	Decision Altitude
DA	Drift Angle
DAL	Development (or Design) Assurance Level
DAPs	Downlink Of Aircraft Parameters
DAR	Designated Airworthiness Representative
DC	Direct Current or Decision Coverage
DCDU	Data Link Control And Display Unit
DCN	Document Change Notice

DCP	Display Control Panel
DD	Dependence Diagram
DDS	Display Docking Station
DER	Designated Engineering Representative
DFMC	Dual Frequency Multiple Constellation
DG	Directional Gyroscope
DGPS	Differential Global Positioning System
DH	Decision Height
DL	Data Link
DLR	Data Link Recorder
DME	Distance Measuring Equipment
DMIR	Designated Manufacturing Inspection Representative
DNC	Direct Noise Canceling
DO	RTCA Document (“Document Order”)
DOD	Department Of Defense
DP	Departure Procedures
DSP	Digital Signal Processor
DUAT	Direct User Access Terminal
DVE	Degraded Visual Environment
EADI	Electronic Attitude Director Indicator
EASA	European Aviation Safety Agency
ECN	Engineering Change Notice
ECU	Engine Control Unit (=EEC)
EEC	Electronic Engine Control
EEPROM	Electrically Erasable Programmable Read Only Memory
EFB	Electronic Flight Bag
EFD	Electronic Flight Display
EFIS	Electronic Flight Information System
EGPWS	Enhanced Ground Proximity Warning System
EGT	Exhaust Gas Temperature
EHS	Enhanced Surveillance
EHSI	Electronic Horizontal Situation Indicator
EIA	Electronics Industry Association

EICAS	Engine Indicating And Crew Alerting System
ELT	Emergency Locator Transmitter
EMI	Electromagnetic Interference
EMS	Emergency Medical Service
ENC	Electronic Noise Canceling
ENG	Engine
ENR	Electronic Noise Reduction
EOC	Executable Object Code
EPR	Engine Pressure Ratio
EPROM	Erasable Programmable Read-Only Memory
ETOP(S)	Extended-Range Twin-Engine Operation(S)
ETSO	European Technical Standard Order
EUROCAE	European Organization for Civil Aviation Equipment
FAA	Federal Aviation Administration
FADEC	Full Authority Digital Engine Control
FAI	First Article Inspection
FANS	Future Air Navigation System
FAR	Federal Aviation Regulation
FBW	Fly-By-Wire
FC	Failure Condition
FCC	Flight Control Computer
FCR	Final Certification Review
FCS	Flight Control System
FD	Flight Director
FDAL	Function Development Assurance Level
FDE	Fault Detection And Exclusion
FDPS	Flight Plan Data Processing System
FDR	Flight Data Recorder
FDRS	Flight Data Recorder System
FDU	Flux Detector Unit
FF	Fuel Flow
FFR	First Flight Review
FFS	Functional Failure Set

FG	Flight Guidance
FHA	Functional Hazard Assessment
FIC	Flight Information Centre
FIFO	“First In, First Out”
FIS	Flight Information Service
FIS-B	Flight Information Services
FL	Flight Level
FLIR	Forward-Looking Infra-Red
FLTA	Forward Looking Terrain Awareness
FM	Formal Methods
FMA	Flight Mode Annunciator
FMEA	Failure Mode & Effects Analysis
FMES	Failure Modes and Effect Summary
FMGS	Flight Management & Guidance System
FMS	Flight Management System
FOB	Fuel On-Board
FPE	Floating Point Emulation
FPGA	Field Programmable Gate Array
FREQ	Frequency
FSS	Flight Service Station
FTA	Fault Tree Analysis
FWS	Flight Warning System
FYDS	Flight Director/ Yaw Damper System
G/S	Glide Slope
GA	General Aviation
GA	Go Around
GAST	GBAS Approach Service Type
GBAS	Ground Based Augmentation System
GCAS	Ground Collision Avoidance System
GCU	Generator Control Unit
GDOP	Geometric Dilution Of Precision
GGs	Global Positioning System Ground Station
GHz	Gigahertz

GLNS	GPS Landing And Navigation System
GLNU	GPS Landing And Navigation Unit
GLONASS	Global Navigation Satellite System
GLS	GBAS Landing System
GLS	GNSS Landing System
GLU	GPS Landing Unit
GMT	Greenwich Mean Time
GND	Ground
GNSS	Global Navigation Satellite System
GO	Go Around
GPIO	General-Purpose I/O
GPS	Global Positioning System
GPWC	Ground Proximity Warning Computer
GPWS	Ground Proximity Warning System
GS	Ground Speed or Ground Station
HAS	Hardware Accomplishment Summary
HAZ	Hazard or Hazardous
HCI	Hardware Configuration Index
HCM	Hardware Configuration Management
HCMP	Hardware Configuration Management Plan
HDG	Heading
HDG SEL	Heading Select
HDL	Hardware Design Language
HDOP	Horizontal Dilution Of Precision
HDP	Hardware Development Plan
HF	High Frequency
HHL D	Heading Hold
HIRF	High Intensity Radiated Field
HMD	Helmet-Mounted Display
HMI	Human Machine Interface
HOL	High-Order Language
HPAP	Hardware Process Assurance Plan
HPR	High Pressure Rotor

HSD	High-Speed Data
HSI	Horizontal Situation Indicator
HSL	Heading Select
HTAWS	Helicopter Terrain Awareness and Warning System
HUD	Head-Up Display
HUMS	Health And Usage Monitoring Systems
HVVP	Hardware Validation & Verification Plan
HW or H/W	Hardware
Hz	Hertz
I/O	Input/Output
IAS	Indicated Airspeed
ICA	Instructions for Continued Airworthiness
ICAO	International Civil Aviation Organization
ICD	Interface Control Document
ICE	In-Circuit Emulator
ID	Identify/Identification Or Identifier
IDAL	Item Development Assurance Level
IDE	Integrated Development Environment
IDENT	Identify/Identifier
IDS	Information Display System Or Integrated Display System
IFE	In-Flight Entertainment
IFF	Identification Friend Or Foe
IFICS	Integrated Flight Instrument And Control System
IFR	Instrument Flight Rules
ILS	Instrument Landing System
IMA	Integrated Modular Avionics
IMC	Instrument Meteorological Conditions
IND	Indicator
INS	Inertial Navigation System
IOC	Initial Operational Capability
IP	Internet Protocol or Intellectual Property
IPV	Instrument Procedure With Vertical Guidance (Renamed to APV)
IRS	Inertial Reference System or Interface Requirements Specification

ISA	International Standard Atmosphere
ISIS	Integrated Standby Instrument System
ISP	Integrated Switching Panel
ISR	Interrupt Service Routine
ITT	Interstage Turbine Temperature
IVSI	Instantaneous Vertical Speed Indicator
JAA	Joint Aviation Authorities
JTAG	Joint Test Action Group
JTIDS	Joint Tactical Information Distribution System
KB	Kilobyte
KIAS	Knots Indicated Airspeed
KT	Knot - Nautical Mile Per Hour
KTAS	Knots True Airspeed
LAAS	Local Area Augmentation System
LADGPS	Local Area Differential GPS
LCD	Liquid Crystal Display
LDGPS	Local Area Differential Global Positioning Satellite
LED	Light-Emitting Diode
LMM	Locator Middle Marker
LOA	Letter Of Agreement/Letter Of Authorization
LOC	Localizer
LODA	Letter of Design Approval
LOI	Level of Involvement
LOM	Locator Outer Marker
LORAN	Long-Range Navigation
LPR	Low Pressure Rotor
LPV	Localizer Performance With Vertical Guidance
LRU	Line Replaceable Unit
LSP	Liskov Substitution Principle
LTE	Loss Of Tail Rotor EffectivenessHelicopters
MA	Markov Analysis
MAP	Manifold Absolute PressureOr Missed Approach Point
MAPS	Minimum Aviation Performance Standards

MASPS	Minimum Aviation System Performance Standard
MB	Megabyte
MB	Marker Beacon
MBD	Model-Based Development
MCBF	Mean Cycles Between Failures
MCDU	Multi-Purpose/Multi-Function Control And Display Unit
MCP	Multi-core Processor, or Multi-core Processing
MC/DC	Modified Condition/Decision Coverage
MDA	Minimum Decent Altitude
MEL	Minimum Equipment List
MF	Medium Frequency
MFD	Multi-Function Display
MFDS	Multi-Function Display System
MIC	Microphone
MIDO	Manufacturing Inspection District Office
MIDS	Multifunctional Information Distribution System
MILSPEC	Military Specification
MKP	Multi-Function Keypad
MKR	Marker Beacon
MLS	Microwave Landing System
MM	Middle Marker
MMD	Moving Map Display
MMEL	Master Minimum Equipment List
MNPS	Minimum Navigation Performance Specifications
MOA	Military Operations Area
Mode A	TransponderPulse-Code Reporting
Mode C	Transponder Code And Altitude Reporting
Mode S	Transponder Code, Altitude, AndTCASReporting
MOPS	Minimum Operational Performance Standard
MOSArt	Modular Open System Architecture
MPS	Minimum Performance Standard
MSA	Minimum Safe Altitude
MSG	Message

MSP	Modes S-Specific Protocol
MSSS	Mode S-Specific Services
MTBF	Mean Time Between Failures
MTBF	Mean Time Between Failures
MTBUR	Mean Time Between Unscheduled Removals
MTTF	Mean Time To Failure
MVA	Minimum Vectoring Altitude
MVER	Marginal Visual Flight Rules
NA	Not Applicable
NACO	National Aeronautical Charting Office
NAS	U.S. National Airspace System
NASA	National Aeronautics And Space Administration
NAV	Navigation Receiver
NAVAID	Navigational Aid
NAVCOMM	Navigation And Communications Equipment Or Receiver
NAVSTAR-GPS	Navigation Satellite Timing And Ranging
NCATT	National Center For Aircraft Technician Training
ND	Navigation Display
NDB	Non-Directional Radio Beacon
NFF	No Fault Found
NFPO	National Flight Procedures Office
NIMA	National Imagery And Mapping Agency
NM or NMI	Nautical Mile
NOAA	National Oceanic And Atmospheric Administration
NoTAM	Notice To Airmen
NPA	Non-Precision Approach
NPRM	Notice Of Proposed Rulemaking
NTAP	Notice To Airmen Publication
NTSB	National Transportation Safety Board
NVD	Night Vision Device
NVG	Night Vision Goggles
NWS	National Weather Service

OAT	Outside Air Temperature
OBS	Omnibearing Selector
OCC	Operations Control Center
ODA	Organization Designation Authorization
OEM	Original Equipment Manufacturer
OM	Outer Marker
OPR	Open Problem Report
OO	Object Oriented
OOT	Object Oriented Technology
OOTIA	Object Oriented Technology In Aviation
OS	Operating System
OWE/OEW	Operating Weight Empty/Operating Empty Weight
PA	Process Assurance
PA	Public Address System
PAL	Pilot Activated Lighting
PAPI	Precision Approach Path Indicator
PAR	Precision Approach Radar
PASA	Preliminary Aircraft Safety Assessment
PC	Personal Computer
PCL	Pilot Controlled Lighting
PCN	Product Change Notice
P-Code	GPS Precision Code
PD	Profile Descent
PDI	Parameter Data Items
PDL	Product Development Leader
PDOP	Position Dilution Of Precision
PDR	Preliminary Design Review
PESA	Passive Electronically Scanned Array
PFD	Primary Flight DisplayOr Primary Flight Director
PFDE	Predicted Fault Detection And Exclusion
PHAC	Plan For Hardware Aspects Of Certification
PLD	Programmable Logic Device
PMA	Parts Manufacturing Approval

PMG	Permanent Magnet Generator
PND	Primary Navigation Display
PNR	Passive Noise Reduction
POA	Production Organization Approval
POF	Phase Of Flight
POH	Pilot's Operating Handbook
POS	Position
PR	Problem Report (may also be "CR = Change Request")
PRA	Pre-Recorded Announcement
PRA	Particular Risk Analysis
P-RNAV	Precision Area Navigation
PSAA	Plan for Software Aspects of Approval
PSAC	Plan For Software Aspects Of Certification
PSCP	Project Specific Certification Plan
PSP	Partnership For Safety Plan
PSR	Primary Surveillance Radar
PSSA	Preliminary System Safety Assessment
PSU	Passenger Service Unit
PTN	Problem Tracking Number
PTR	Program Trouble Report
PTT	Push-To-Talk
QA	Quality Assurance
QAR	Quick Access Recorder
QM	Quality Management
QNH	Barometric Pressure Adjusted To Sea Level
QRH	Quick Reference Handbook
RA	Resolution Advisory (TCAS)
RA	Resolution Advisory (Traffic Function)
Rad Alt	Radio Altitude
RAI	Radio Altimeter Indicator
RAIM	Receiver-Autonomous Integrity Monitoring
RALT	RadarOrRadio Altimeter
RAM	Random Access Memory

RAT	Ram Air Turbine
RCR	Reverse Current Relay
RCVR	Receiver
RDMI	Radio Distance Magnetic Indicator
RDP	Radar Data Processing System
RDR	Radar
RDU	Remote Display Unite
REF	Reference
REIL	Runway End Identifier Lights
REL	Relative
RF	Radio Frequency
RFI	Radio Frequency Interference
RHSM	Reduced Horizontal Separation Minimal
RJ	Regional Jet
RLG	Ring Laser Gyroscope
RLY	Relay
RMI	Radio Magnetic Indicator
R-NAV	Area Navigation
RNG	Range
RNP	Required Navigation Performance
ROC	Rate Of Climb
ROD	Rate Of Descent
ROM	Read Only Memory
RPA	Remotely Piloted Aircraft (Unmanned Aerial Vehicle)
RPM	Revolutions Per Minute
RSP	Reversion Switch Panel
RTE	Route
RTL	Run Time Library
RTOS	Real-Time Operating System
RVR	Runway Visual Range
RVSM	Reduced Vertical Separation Minimum
RX	Receiver
SAAR	Special Aircraft And Aircrew Requirements

SAE	Society Of Automotive Engineers
SAR	Search And Rescue, Smart ACMS Recorder
SAS	Software Accomplishment Summary
SAT	Static Air Temperature
SATCOM	Satellite Communication
SATNAV	Satellite Navigation
SATNAV	Satellite Navigation
SC	System Control Category
SCI	Software Configuration Index (or “Item”)
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SCR	Software Conformity Review
SCS	Software Coding Standard
SCS	System Control Unit
SD	Secure Digital
SDD	Software Design Description
SDF	Simplified Directional Facility
SDP	Software Development Plan
SDRL	Supplier Deliverables
SDS	Software Design Standard
SECI	Software lifecycle Environment Configuration Index
SELCAL	Selective Calling
SES	Supplier Equipment Specification
SID	Standard Instrument Departure
SIU	Satellite Interface Unit
SLA	Software Load Application
SMS	Short Messaging Service
SNR	Signal-To-Noise Ratio
SOI	Stage Of Involvement
SOP	Standard Operating Procedure
SOUP	Software of Unknown Pedigree (sometimes used with humor ...)
SOW	Statement Of Work
SPP	Safety Program Plan

SPR	Software Planning Review
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SQAR	Software Quality Assurance Representative
SRATS	System Requirements Allocated to Software
SRD	Software Requirements Data
SRS	Software Requirements Standards
SRS	Speed Reference System or Software Requirements Standard
SSA	System Safety Assessment
SSCV/DR	Solid-State Cockpit Voice/Data Recorder
SSCVR	Solid-State Cockpit Voice Recorder
SSFDR	Solid-State Flight Data Recorder
SSPP	System Safety Program Plan
SSR	Secondary Surveillance Radar
SSR	System Specification Review; alternatively: Software Specification Review
SSS	System Segment Specification
STA	Static Timing Analysis
STAR	Standard Terminal Arrival Routes
STAR	Standard Terminal Arrival Route
STARS	Standard Terminal Automation Replacement System
STC	Supplemental Type Certificate
STCA	Short-Term Conflict Alert
STP	Software Test Protocol
STP	Standard Temperature And Pressure
STS	Software Test Specification
SUA	Special Use Airspace
SVA	Software Validation Attestation
SVCP	Software Verification Cases And Procedures
SVP	Software Verification Plan
SVR	Software Verification Results
SW or S/W	Software
SWCEH	Software and Complex Electronic Hardware
SYRD	System Requirements Document

T/R	Thrust Reverser Or Tail Rotor
TA	Traffic Advisory
TACAN	Tactical Air Navigation System
Tach	Tachometer
TAD	Terrain Awareness Display
TAF	Terminal Area Forecast
TAS	True Airspeed or Tool Accomplishment Summary
TAT	True Air Temperature, Or Total Air Temperature
TAWS	Terrain Awareness and Warning System
TBD	To Be Defined (or Determined; interchangeable)
TBO	Time Before Overhaul, Or Time Between Overhaul
TC	Type Certificate or Test Case
TCA	Throttle Control Assembly, Or Terminal Control Area
TCAD	Traffic Collision Alert Device
TCAS	Traffic Collision Alert System or Traffic Collision Avoidance System
TCF	Terrain Clearance Floor
TCI	Tool Configuration Index
TCN	TACAN
TCR	Test Completeness Review
TCU	TACAN Control Unit
TDOP	Time Dilution Of Precision
TDR	Transponder
TERPS	Terminal En-Route Procedures
TFR	Temporary Flight Restrictions
TFT	Thin-Film Transistor
TGT	Turbine Gas Temperature, Or Target
THDG	True Heading
TIS	Traffic Information Service
TK	Track Angle
TKE	Track-Angle Error
TLA	Thrust Lever Angle
TOD	Top Of Descent Point
TOR	Tool Operational Requirements

TQ	Tool Qualification
TQL	Tool Qualification Level
TQP	Tool Qualification Plan
TR or T/R	Transmitter Receiver OrTransceiver
TRACON	Terminal Radar Approach Control
TRANS	Transmit,Transmission, OrTransition
TRK	Track
TRP	Mode S Transponder
TRR	Test Readiness Review
TSO	Technical Standard Order
TTL	Tuned To Localizer
TTR	TCAS II Transmitter/Receiver
TTS	Time To Station
TVE	Total Vertical Error
TWDL	Two-Way Data Link, Or Terminal Weather Data Link
TWDR	Terminal Doppler Weather Radar
TWIP	Terminal Weather Information For Pilots
TWR	Terminal Weather Radar
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UHF	Ultra-High Frequency
ULB	Underwater Locator Beacon
UML	Unified Modified Language
USAF	United States Air Force
USB	Universal Serial Bus
USGS	United States Geological Survey
UTC	Universal Time Coordinate
V	VoltsOrVoltage
V&V	Validation and Verification
V/L	VOR/Localizer
V/NAV	Vertical Navigation

V/R	Voltage Regulator
V/REF	Reference Velocity
V/S	Vertical Speed
V/TRK	Vertical Track
VASI	Visual Approach Slope Indicator
VASIS	Visual Approach Slope Indicator(System)
VDF	VHF Direction Finding
VDL	VHF Data Link
VDR	VHF Digital Radio
VFO	Variable Frequency Oscillator
VFR	Visual Flight Rules
VG/DG	Vertical Gyroscope/Directional Gyroscope
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language (VHSIC = Very High Speed Integrated Circuit)
VHF	Very High Frequency
VMC	Visual Meteorological Conditions
VMC	Visual Meteorological Conditions
VNE	Never Exceed Speed
VNO	Maximum Structural Cruising Speed
VNR	VHF Navigation Receiver
VOR	VHF Omnidirectional RangeAnd Ranging
VOR/DME	VORWithDistance Measuring Equipment
VOR/MB	VOR Marker Beacon
VORTAC	VOR And TACAN Combination
VOX	Voice Transmission
VPA	Vertical Path Approach
VPATH	Vertical Path
VRP	Visual Point Of Reference
VSI	Vertical Speed Indicator
VSM	Vertical Separation Limit
VSO	Stall SpeedIn Landing Configuration
VSWR	Voltage

VX	Speed For Best Angle Of Climb
VY	Speed For Best Rate Of Climb
WAAS	Wide Area Augmentation System
WBS	Work Breakdown Structure
WCET	Worst-Case Execution Time
WD	Wind Direction
WINDR	Wind Direction
WMA	WXR Waveguide Adapter
WMI	WXR Indicator Mount
WMS	Wide-Area Master Station
WMSC	Weather Message Switching Center
WMSCR	Weather Message Switching Center Replacement
WPT	Waypoint
WRT	WXR Receiver Transmitter
WS	Wind Shear
WX	Weather
WXR	Weather Radar System
WYPT	Waypoint
XCVR	Transceiver
XFR	Transfer
XMIT	Transmit
XMSN	Transmission
XMTR	Transmitter
XPDR	Transponder
XTK	Crosstrack
ZSA	Zonal Safety Analysis

The Avionics Development Ecosystem

Applying DO-178C & Related Guidelines

By
Vance Hilderman
With 10+ Industry Experts



Afuzion Press

Appendix B:

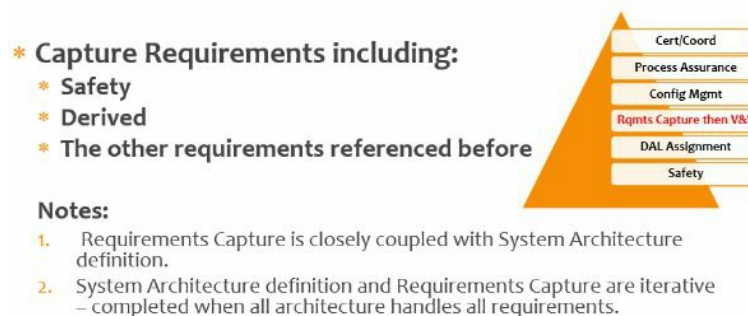
Aviation System Requirements Checklist

Summary:

The “System Requirements Data Checklist” ensures that System-level requirements (including Safety and Derived requirements) are properly defined. There should be a System Requirements Standard and also compliance with ARP4754A and ARP4761/A. Aviation development companies either buy or build their own checklists covering all aspects of the aviation engineering lifecycle. To find commercially available checklists, a quite common approach is to enter “ARP4754A Requirements Checklists” in an internet search engine query and view the results.

Background:

In avionics systems, requirements provide the foundation. Requirements must be correct, feasible, uniquely identified, traced, decomposable to hardware and software requirements, verified and validated. This checklist should be customized for your particular project to ensure ARP-4754A is adhered to. It should be noted that System Requirements definition is closely coupled (performed in parallel) with system architectural definition. Customer, external, environmental (DO-160 compliant) and FHA-related Safety requirements are all considered to compose a draft system architecture inclusive of monitoring and redundancy. This proposed architecture is then iteratively refined inclusive of system requirement evolution until all system requirements are defined. The graphics below depict this process:





Supplier Name:	Location:
System/LRU Name:	System/LRU Identifier:
Data Item Being Reviewed (Full Artifact Name):	
Data Item Revision:	Applicable Standards: ARP-4754A, ARP-4761(A), DO-160, DO-326A (Cyber-Security)?,
Review Date:	Principal Reviewer Name:
Additional Reviewer Names (Optional):	

Instructions:

Review the document/artifact for compliance against the requirements identified in each checklist item. Mark the “Yes/No” column as follows:

- **“YES”** if the artifact complies with the checklist item.
- **“NO”** if the artifact does not comply with the checklist item. Include comments and/or reference to detailed comments in the “Remarks” column.
- **“N/A”** if the checklist item is not applicable and therefore requires no review. For instance, if an item applies to only Levels A – C, and the system is for Level D System. Include a rationale for “N/A” in the “Remarks” column.

Note: All non-compliance issues must be resolved before the document can be approved.

1.0 General Checklist Content:

Criteria	Yes or No	Remarks
a. Has this document been reviewed by the submitting project’s applicable certification authority?	Y N	
b. Has this document been reviewed by the submitting project’s applicable Process Assurance representative?	Y N	

c. Is this artifact (document) under configuration control as defined in the referenced Configuration Management Plan?	Y N	
d. Does the document identify the system name, release date, and document identification and revision level?	Y N	
e. Are the applicable terms and acronyms defined for each acronym used within the document?	Y N	
f. Is the referenced document section complete and with all applicable project documents referenced or associated with certification/engineering criteria?	Y N	

2.0 Specific Checklist Content:

Query	Compliant	Remarks
System Requirements		
Have applicable external environmental requirements been allocated to the system, e.g. from DO-160?	YES NO NA	
Have applicable external performance requirements been allocated to the system, e.g. from applicable TSO's?	YES NO NA	
Have potential failures and outputs of the FHA process been incorporated into safety-related System requirements?	YES NO NA	
Are the system requirements consistent with the system-level architecture?	YES NO NA	
Is there a description of the allocation of system requirements to hardware or software, with attention to safety-related requirements and potential failure conditions?	YES NO NA	
Have fault conditions identified in the FHA been addressed via Safety requirements allocated to the System?	YES NO NA	
Do the System requirements conform to the System requirements standard, ARP4754A, for safety requirements, also ARP-4761(A)?	YES NO	
If a requirement deviates from the Requirements Standard, is the deviation independently assessed prior to acceptance?	YES NO	
Are the System requirements:		

unambiguous?	YES NO	
consistent?	YES NO	
complete, leaving no undefined conditions?	YES NO	
verifiable by analysis or test? Examples of unverifiable terms: “ flexible, easy, sufficient, safe, ad hoc, adequate, accommodate, user-friendly, usable, when required, if required, appropriate, fast, portable, light-weight, small, large, maximize, minimize, sufficient, robust, quickly, easily, clearly, other “ly” words, other “ize” words”)	YES NO	
stated in quantitative terms with tolerances, where applicable?	YES NO	
Is the requirement stated positively (as opposed to negatively, i.e., “shall not”)?	YES NO	
Is each requirement unique to avoid redundancy in whole or in part?	YES NO	
Do the System requirements not describe hardware/software design or verification detail, except for specified and justified design constraints?	YES NO	
Are System requirements devoid of requirements better relegated to Software High-Level requirements and/or Hardware requirements?	YES NO	
Are System requirements which identify and preclude system hazards defined?	YES NO N/A	
If applicable, do the System requirements describe how any parameter data item (PDI) (a superset of configuration data) is used by the System, including:		
PDI structure?	YES NO N/A	
PDI attributes?	YES NO N/A	
when applicable, values that are consistent with the structure of the PDI and the attributes of its data elements?	YES NO N/A	
Are the functional and operational requirements defined for each mode of operation?	YES NO N/A	
Do the System requirements address performance criteria (e.g., precision and accuracy)?	YES NO N/A	
Are memory size constraints defined?	YES NO N/A	
Are timing requirements and constraints defined?	YES NO N/A	

Do the System requirements address hardware and System interfaces, including:		
protocols?	YES NO N/A	
formats?	YES NO N/A	
frequency of inputs?	YES NO N/A	
frequency of outputs?	YES NO N/A	
Are the failure detection requirements defined?	YES NO N/A	
Are the safety monitoring requirements defined, and compliant with ARP4754A and ARP4761/A?	YES NO N/A	
If the system includes requirements for partitioning, are the following items addressed:		
are the partitioning requirements that are allocated to the System defined?	YES NO N/A	
is there a description of how the partitioned System components interact with each other?	YES NO N/A	
is the System level of each partition defined?	YES NO N/A	
Are requirements for built-in testing and monitoring defined, including detection, isolation, reporting, health monitoring, and switchover?	YES NO N/A	
Are requirements for recovery from and System failures defined?	YES NO N/A	
Are requirements for recovery from abnormal conditions defined, including auto-restart and power-fail and transients?	YES NO N/A	
Does the System requirements data include a requirement for the System/Software configuration item identifier to be embedded in the system/software and defined so that the identifier can be observed, accessed, or otherwise determined?	YES NO N/A	
Have derived requirements been provided to the to the system safety assessment process?	YES NO N/A	
If applicable, are requirements for System Redundancy complete including details on conditions which may invoke switchover, timing, and determination of health determination including voting (particularly for DAL A or B systems)	YES NO N/A	
Do the requirements ensure that there is independence of functions and their monitors by requiring that the monitor and its protective	YES NO N/A	

mechanism are not rendered inoperative by the same failure condition that causes the hazard?		
If the system includes requirements for user-modifiable data, are the following items addressed:		
Are there requirements which specify the mechanisms which prevent the user modification from affecting systems safety whether or not they are correctly implemented?	YES NO N/A	
Is the capability which provides the protection for user modification at the same System level as the function it is protecting from errors in the modifiable component?	YES NO N/A	
Is the user not allowed to modify the System unless compliance with ARP-4754A is demonstrated for the modification?	YES NO N/A	
Are there provisions for the user to take responsibility for all aspects of the user-modifiable System at the time of the user modification, for example, System configuration management, System quality assurance, and System verification?	YES NO N/A	
If the system includes requirements for option-selectable software or hardware, is the following item addressed:		
When System programmed options or configurations are included, are there means provided to ensure that inadvertent selections involving non-approved configurations for the target computer within the installation environment cannot be made?	YES NO N/A	
If the system has a default mode when inappropriate System or data is loaded, then does each partitioned component of the system have safety-related requirements specified for operation in this mode which address the potential failure condition?	YES NO N/A	
Are requirements for Installation present and in compliance with the safety assessment (particularly for Common Mode Analysis)?	YES NO N/A	
Are requirements for operation present and in compliance with the safety assessment?	YES NO N/A	
Are requirements for maintenance present and in compliance with the safety assessment?	YES NO N/A	
If System includes an airborne display mechanism	YES NO N/A	

that is the means for ensuring that the aircraft conforms to a certified configuration, then is the System developed to the highest level of the System to be loaded or does the system safety assessment process justify the integrity of an end-to-end check of the System configuration identification?		
If the system has a loading function, including support systems and procedures, is there a means defined to detect incorrect software and/or hardware and/or aircraft combinations and is there protection provided appropriate to the failure condition of the function?	YES NO N/A	
System Requirements		
Is traceability between system requirements and safety requirements provided to enable verification of the complete implementation of the system requirements and give visibility to the derived requirements that are not directly traceable to system requirements?	YES NO N/A	
Is each system requirement able to be decomposed into hardware or software high level requirements?	YES NO N/A	
Can each system requirement be validated and verified by one or more test cases?	YES NO N/A	

Comment Section: Include additional review comments, a summary of observations, and necessary actions required to accept this document.

Additional Review Comments:

- a.
- b.
- c.

Summary of Observations:

- a.
- b.
- c.

Necessary Actions required for accepting this document:

- a.
- b.
- c.